MARCOS ANTONIO MASNIK FERREIRA

MÉTODOS PARA GERAÇÃO DE ENTIDADES, EM MODELOS DE SIMULAÇÃO, PARA PROCESSOS ESTOCÁSTICOS DE RENOVAÇÃO NÃO ESTACIONÁRIOS

Tese apresentada como requisito parcial à obtenção do grau de Doutor em Ciências pelo Programa de Pós-Graduação em Métodos Numéricos em Engenharia, Setor de Ciências Exatas e Setor de Tecnologia, Universidade Federal do Paraná.

Orientador:

Prof. Dr. Celso Carnieri - UFPR

Coorientador:

Prof. Dr. Rui Carlos Botter - USP

2009

TERMO DE APROVAÇÃO

MARCOS ANTONIO MASNIK FERREIRA

MÉTODOS PARA GERAÇÃO DE ENTIDADES, EM MODELOS DE SIMULAÇÃO, PARA PROCESSOS ESTOCÁSTICOS DE RENOVAÇÃO NÃO ESTACIONÁRIOS

Tese aprovada como requisito parcial para a obtenção do grau de Doutor em Ciências, na área de concentração em Programação Matemática do Programa de Pós-Graduação em Métodos Numéricos em Engenharia da Universidade Federal do Paraná, pela seguinte banca examinadora:

Orientador:

Celso Carnieri, Dr.
PPGMNE, UFPR

Coorientador:

Rui Carlos Botter, Dr.
Escola Politécnica, USP

Joel Maurício Corrêa da Rosa, Dr.
Instituto de Matemática, UFF

Arinei Carlos Lindbeck da Silva, Dr.
Engenharia de Produção, UFPR

Neida Maria Patias Volpi, Drª.
Engenharia de Produção, UFPR

Curitiba, 29 de junho de 2009

AGRADECIMENTOS

À Silvana, Leandro, Bruna e Juliana pelo convívio e apoio durante estes anos para a concretização do trabalho.

Aos meus pais (Estefanida e Nelson) pela sua luta, apoio e carinho.

Ao orientador Dr. Celso Carnieri pela sua condução e orientação do trabalho e, fundamentalmente, pelo seu comportamento humano ao longo desta jornada.

Ao coorientador Dr. Rui Carlos Botter pelas suas inestimáveis contribuições, críticas e sugestões.

Ao Estado Brasileiro, representados pela Universidade Federal do Paraná e Banco Central do Brasil, que deram a estrutura necessária, sem a qual seria muito difícil a concretização da tese.

‘SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

DRP – Delayed Renewal Process

ESBC – Engenharia de Software Baseada em Componentes

fp – Função de probabilidade

fdp – Função Densidade de Probabilidade

FDC – Função de Distribuição Cumulativa

FDCE – Função de Distribuição Cumulativa Empírica

GNA – Gerador de Números Aleatórios

GCL – Gerador Congruencial Linear

GRMC – Gerador Recursivo Multiplicativo Combinado

HW – Half Width (meia largura do Intervalo de Confiança)

IID – Independente e Identicamente Distribuído

IC – Intervalo de Confiança

MT – Mecanismo de Transporte

PPBP – Poisson Pareto Burst Process

PPLI – Problema de Programação Linear Inteira

PPLIM – Problema de Programação Linear Inteira Mista

PR – Processo de Renovação

TP – Terminal Portuário

RESUMO

Em modelos de simulação para determinados processos de renovação (PR), as funções de geração de sequências aleatórias são insuficientes para se modelar fielmente a chegada das entidades no modelo desenvolvido, visto que esses processos sofrem a influência simultânea de fatores aleatórios e determinísticos. Para que a simulação possa ser escolhida como técnica de análise para esses processos e até ter seu uso ampliado, é necessário, portanto, superar essas dificuldades, já que o elevado tempo de desenvolvimento não só pode inibir o uso de simulação, mas também apresentar um custo proibitivo para o projeto. Considerando que a ampla adoção da técnica de simulação para a análise de sistemas complexos exige que os projetistas disponham de um ambiente com elevado nível de abstração, liberando-os de tarefas que não sejam exclusivamente relacionadas à complexidade do problema em estudo, esta pesquisa concentra-se nas fases de *Coleta de Dados* e *Implementação Computacional*, tidas como as mais demoradas e caras na construção de modelos. A pesquisa apresenta e propõe métodos para a geração de entidades, em ambientes de simulação, para processos estocásticos não estacionários, incorporando todas as relações matemáticas e estatísticas indispensáveis para a geração de entidades e, por conseguinte, liberando o projetista dessas complexidades. Este trabalho apresenta todos os algoritmos e equações que implementam a solução, usando Programação Linear Inteira Mista (PLIM), programação por objetivos e um algoritmo para a obtenção de uma sequência com quantidade fixa de eventos limitados dentro de um intervalo e de acordo com uma distribuição de probabilidade previamente determinada. Mostra, igualmente, como os métodos propostos podem ser usados em sistemas reais e complexos nas áreas de administração, redes de computadores e logística, com o propósito de demonstrar a sua utilidade e, consequentemente, sua contribuição científica.

ABSTRACT

In simulation models developed to some renewal process (RP), the available random deviate functions are not able to properly model the entities arrivals, since these processes have stochastic and deterministic elements. In order to simulation may be chosen as the appropriate technique for the analysis of these processes and also for the purpose of achieving a large-scale adoption, it is necessary to overcome these problems, since its time-consuming development may not only avoid its use but also present prohibitive costs to the project. Considering that a large-scale simulation adoption to the analysis of complex systems depends on the availability of development environments with high abstraction levels, hiding implementation details from simulation modelers, allowing them to worry only about the model's complexities, this research focuses on the *Data Collection* and *Computer Translation* simulation model steps, which are the ones presenting the higher costs and time-consuming levels. This research presents and proposes numerical methods to generate random deviates, in simulation environments, for nonstationary stochastic process, incorporating all the necessary mathematics and statistical relations, relieving the simulation builders of these complexities. In addition, this research presents all the algorithms and equations that implement the solution by means of Mixed-Integer Linear Programming (MILP), goal programming and an algorithm to get a random sequence with a fixed number of events bounded into an interval fitted to a specified probability distribution. These methods are applied to complex real-world systems in the Management Science, computer networks and logistics, aiming to demonstrate its usability and, consequently, its scientific contribution.

1 INTRODUÇÃO

A simulação destaca-se como uma das principais técnicas da Pesquisa Operacional sendo indispensável para a análise de sistemas, segundo Bardos (1999), quando as seguintes características estiverem presentes:

- O modelo é complexo, com muitas variáveis, e seus componentes interagem;
- A relação entre as variáveis do problema é não-linear;
- O problema inclui variáveis aleatórias como intervalos de chegada dos clientes, tempos de serviço, falhas em equipamentos que ocorrem de acordo com uma distribuição de probabilidades, etc.;
- Uma apresentação visual do comportamento do sistema é necessária para a compreensão dos resultados por parte dos usuários. Além disso, a visualização do funcionamento do sistema pelos usuários aumenta a credibilidade dos resultados apresentados.

A técnica de simulação refere-se a um amplo conjunto de métodos e aplicações com a finalidade de imitar ou simular o comportamento de sistemas reais, usualmente implementada em ambientes computacionais com softwares apropriados (KELTON; SADOWSKI; SADOWSKI, 2001). Segundo Chwif (1999), a crescente popularidade da simulação se deve basicamente aos seguintes fatores:

- Desenvolvimento da indústria de computadores;
- Desenvolvimento e aprimoramento dos softwares de simulação, com interfaces mais “amigáveis” e avanços de processamento gráfico;
- Capacidade desta técnica de modelar sistemas complexos, sendo especialmente importante quando não há uma solução analítica (por meio de modelos matemáticos) para o problema.

Um **modelo** pode ser entendido como uma representação de um sistema real e, assim, uma questão pertinente a esta representação é quais seriam os limites do modelo que supostamente melhor representam o sistema. Como regra geral, o modelo deve ser complexo apenas, e tão somente, o necessário a fim de atender aos seus objetivos (BANKS, 1998). Um **evento** é uma ocorrência que modifica o **estado** do sistema. Existem eventos externos e internos, também chamados de eventos exógenos e endógenos, respectivamente. Um evento endógeno pode ser o término de atendimento de cliente e um evento exógeno a chegada de um cliente no sistema.

Os **modelos de simulação** podem ser classificados de acordo com o estado do sistema e a presença ou não de aleatoriedade nos seus dados de entrada.

Nance (1993) apresenta uma taxonomia que divide os modelos de simulação em três partições levando em consideração o estado do sistema: **simulação contínua**, **simulação de eventos discretos** e **simulação Monte Carlo**. Caso o estado mude continuamente com o avanço do tempo, então o modelo é classificado como **contínuo**. A simulação contínua é adequada para modelar fenômenos físicos com o uso de equações diferenciais como o embarque dos porões de um navio, a temperatura de uma caldeira em uma siderúrgica ou o preço de uma opção em um determinado tempo *t*, pois os valores das variáveis de interesse nestes casos são alterados continuamente à medida que a simulação avança. Quando o modelo é **discreto**, as mudanças de estado ocorrem em pontos específicos de tempo. Chegadas e partidas de clientes ou pedidos em um sistema e interrupções no funcionamento de máquinas ou equipamentos devido a falhas são exemplos de eventos que ocorrem somente em determinados instantes da simulação sendo, portanto, eventos ditos discretos. Ressalte-se, entretanto, que em modelos mais sofisticados de simulação é muito comum a ocorrência simultânea de eventos contínuos e discretos. **Monte Carlo** é um método de simulação no qual um problema de difícil solução, notadamente determinístico, é resolvido por meio de um processo estocástico que satisfaça as relações do problema. Diferentemente das simulações contínua e discreta, a representação do tempo não é necessária na simulação Monte Carlo. Um exemplo típico de solução por este método é a solução de integrais definidas circunscrevendo a região da função com uma figura geométrica conhecida e, então, gerando pontos aleatórios dentro desta região. Ao final de um grande número de iterações, calcula-se a proporção de pontos que caíram dentro dos limites da função com os pontos externos. Esta proporção estabelece a região limitada pela função.

Modelos que não possuem variáveis aleatórias são classificados como **determinísticos** como, por exemplo, modelos cuja chegada de clientes se dá em tempos preestabelecidos ou cujos tempos de serviços são fixos. Para estes modelos, o resultado é sempre o mesmo, não importando quantas rodadas de simulação sejam efetuadas. Por outro lado, modelos **estocásticos** são construídos utilizando variáveis que seguem um comportamento aleatório – intervalos de chegada de clientes no sistema ou tempos de serviço são definidos de acordo com uma distribuição de probabilidades.

Apesar do crescimento verificado na modelagem de sistemas complexos usando simulação, as linguagens e os ambientes de desenvolvimento precisam superar os seguintes desafios:

- O tempo despendido no projeto e na construção de modelos, normalmente, é elevado e, em consequência, os custos dos projetos são altos;
- Um projeto de simulação requer profissionais altamente qualificados.

Comparado com os métodos analíticos, a simulação é mais trabalhosa e é especialmente caro o desenvolvimento de modelos bastante detalhados. Normalmente, o nível deste detalhamento é limitado ou por questões orçamentárias ou pelo prazo de entrega do projeto. Para a superação destes obstáculos ou, pelo menos, a minimização destes problemas, é necessário que as ferramentas de simulação sejam suficientemente flexíveis e eficientes, apresentando aos projetistas componentes com elevado nível de abstração.

Giunchiglia,Villafiorita e Walsh (1997) definem abstração como sendo um mapeamento de uma representação do problema para outra, que preserva certas propriedades desejáveis e reduz sua complexidade. Pode-se dizer que, basicamente, há dois níveis de abstração: abstração no controle e abstração nos dados. A abstração no controle existe quando se provê aos usuários de um determinado sistema um conjunto de ferramentas que permitem a solução de um problema mais complexo, escondendo-se detalhes de implementação. A abstração nos dados ocorre nos ambientes computacionais por meio dos sistemas operacionais ou linguagens de programação que apresentam aos usuários representações de dados (estruturas de dados, arquivos, etc), escondendo a representação subjacente nestes ambientes.

Vliet (2007, p. 117) discute a taxonomia para os atributos de qualidade de software baseados, principalmente, no trabalho de (McCALL; RICHARDS; WALTERS, 1977) e no padrão ISO 9126 (ISO9126, 2001). Dentre os principais atributos comuns a ambos os trabalhos, destacam-se:

- **Corretude**: o quanto o software satisfaz a sua especificação e ao atendimento de seus objetivos;
- **Confiabilidade**: em que extensão o software realiza sua função com a precisão requerida;
- **Eficiência**: a quantidade de recursos computacionais e o código necessário para o software realizar sua função;
- **Integridade**: qual é a extensão do controle de acesso ao software por pessoas não autorizadas;
- **Usabilidade**: o esforço necessário para aprender, operar, preparar dados de entrada e interpretar a saída do software;
- **Manutenibilidade**: o esforço necessário para localizar e corrigir um erro no software;
- **Testabilidade**: o esforço necessário para assegurar que o software realize corretamente sua função;
- **Flexibilidade**: o esforço necessário para modificar o software;
- **Portabilidade**: o esforço necessário para transferir um software de um hardware ou ambiente operacional para outro;
- **Interoperabilidade**: o esforço para juntar um sistema ou software a outro.

Presente em (McCALL; RICHARDS; WALTERS, 1977), mas ausente do padrão ISO, destaca-se ainda o atributo de qualidade **Reusabilidade** (a extensão de quanto um software pode ser reutilizado em outras aplicações). Isto decorre do fato de que o padrão ISO

refere-se estritamente à avaliação de um *produto* final e não ao *processo* de desenvolvimento de software.

O que se pretende com este trabalho de pesquisa é mitigar os dois problemas no desenvolvimento de projetos de simulação (elevado tempo despendido num projeto de simulação e o requisito de profissionais altamente qualificados), provendo ferramentas de geração de entidades para processos estocásticos de renovação não estacionários, liberando os projetistas destas complexidades de forma que esses possam se preocupar exclusivamente com as questões relativas aos seus modelos. Desta forma, não só o processo de desenvolvimento de modelos pode ser acelerado, bem como o tempo de manutenção de modelos poderá ser reduzido.

Provendo-se aos projetistas de modelos de simulação, ferramentas que gerem automaticamente entidades para processos estocásticos de renovação não estacionários, não só se alcança os objetivos da redução de tempo de desenvolvimento de modelos com menor necessidade de qualificação profissional, mas simultaneamente há um ganho de qualidade nos modelos desenvolvidos, considerando os atributos acima citados.

1.1 ETAPAS DE UM PROJETO DE SIMULAÇÃO

Banks (1998, p. 15) e Musselman (1998) apresentam os passos que devem guiar um projetista de simulação durante o desenvolvimento de um projeto de simulação, conforme mostra a Figura 1.1.

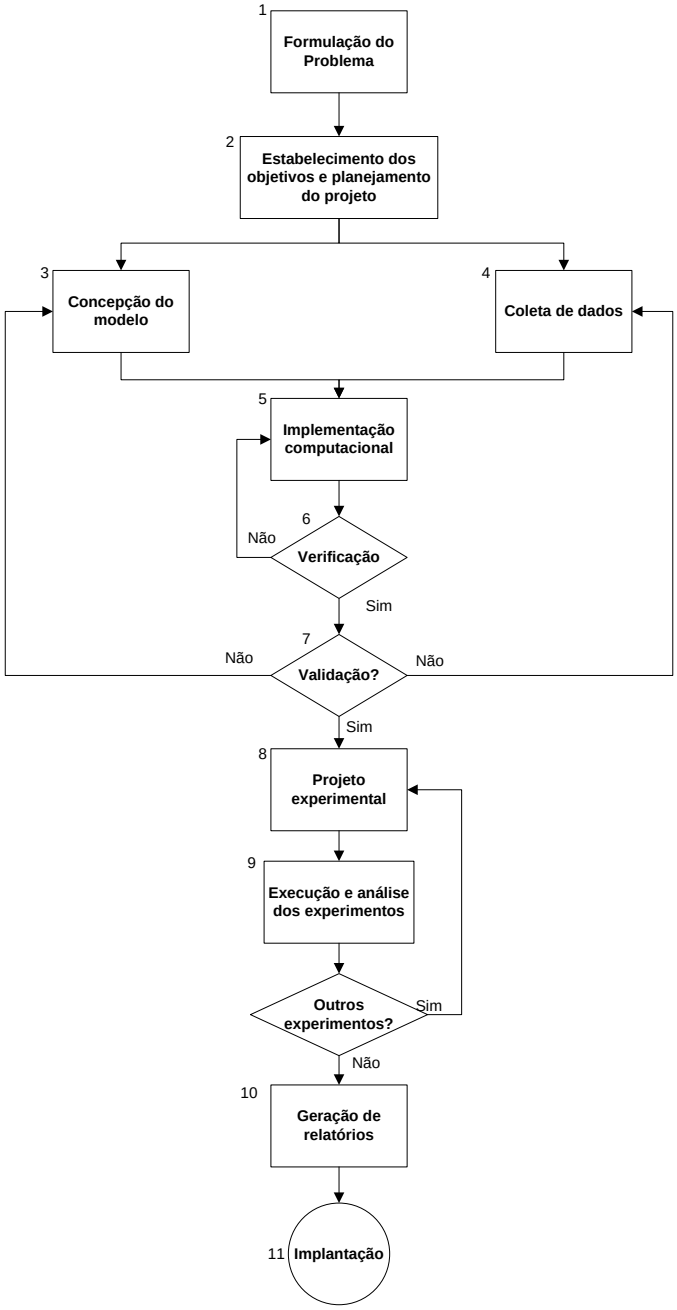

Figura 1.1- Etapas de um projeto de simulação

Fonte: (BANKS, 1998).

1. **Formulação do problema**. Ao se iniciar um projeto de simulação, é de fundamental importância que o problema a ser resolvido esteja claramente definido tanto para o projetista quanto para o cliente. Este deveria ser um princípio a ser seguido qualquer que fosse a técnica de modelagem escolhida, mas quando simulação é a técnica a ser utilizada o impacto de se levar adiante

um projeto mal-definido implicará, sem dúvida, custos extremamente elevados. Esta etapa vai orientar todas as demais atividades do projeto e, neste ponto, o projetista deve estabelecer as questões centrais e definir o escopo do projeto. Ainda que esta etapa tenha sido tratada com extrema cautela, é possível que seja necessário reavaliar estas definições à medida que se avance no desenvolvimento do projeto.

2. **Estabelecimento dos objetivos e planejamento do projeto**. Os objetivos definem quais são as questões chave do problema que precisam ser respondidas pelo projeto: listar essas questões, classificá-las por ordem de prioridade, definir os relatórios que serão gerados, identificar os dados de entrada, entre outras são algumas tarefas desta fase. O planejamento deve definir quais cenários serão investigados e, além disso, indicar os recursos necessários (pessoal, hardware e software) para o desenvolvimento do trabalho.

3. **Concepção do modelo**. O modelo conceitual é uma abstração do sistema real, sendo composto pelos elementos do sistema, suas características e interações, estabelecidas por uma série de relações matemáticas, estatísticas e lógicas. Recomenda-se que se inicie com um modelo simples e que, paulatinamente, vá se adicionando complexidade até o ponto necessário a fim de que os objetivos definidos na fase anterior sejam alcançados.

4. **Coleta de dados**. Nesta fase todos os dados necessários para a construção do modelo devem ser obtidos junto ao cliente do projeto. Esta é uma das fases mais difíceis, principalmente, quando se consideram os modelos estocásticos de simulação. É necessário obter dados na quantidade suficiente e de qualidade para que se possa fazer uma análise razoável. Em muitos contextos, a obtenção de dados é absolutamente impossível - considere um projeto de simulação para um sistema que ainda não está em execução. Para outros projetos, é possível que não haja tempo suficiente para que se faça uma análise detalhada e aprofundada dos dados de entrada do sistema. A fase de concepção do modelo é desenvolvida concomitantemente com a fase da coleta de dados. Isto significa que o modelo conceitual é criado num processo interativo com a obtenção dos dados.

5. **Implementação computacional**. Nesta fase o modelo conceitual do sistema (etapa 3) é implementado computacionalmente, podendo ser usado desde uma linguagem de programação de uso geral até um pacote específico de simulação.

6. **Verificação**. A verificação tem por objetivo determinar se o modelo implementado está operando corretamente. O processo da verificação é contínuo e não se deve esperar o término da implementação para se iniciar a fase de verificação.

7. **Validação**[1]. A validação tem por objetivo determinar se o modelo conceitual representa de forma precisa o sistema real. O objetivo desta fase é responder se o modelo pode substituir o sistema real para que se possa realizar experimentos ou análise de cenários.

8. **Projeto experimental**. Para cada experimento ou cenário a ser analisado, é preciso definir a duração da simulação, quantas rodadas ou replicações (*replications)* serão executadas a fim de que se obtenham os valores médios de interesse com os respectivos intervalos de confiança - IC (ver item 2.2)**.**

9. **Execução e análise dos experimentos**. Nesta fase os experimentos são executados por meio do modelo desenvolvido, bem como é feita a análise dos resultados.

10. **Geração de relatórios**. Considerando que os resultados obtidos são válidos e que não há necessidade de se testar outros cenários, os resultados devem ser apresentados em um relatório conciso e claro. Além disso, a documentação do

[1] Balci (1998) é excelente referência para as etapas de verificação e validação. Este autor defende que o ciclo de vida de um projeto de simulação não deve ser visto como uma atividade estritamente seqüencial. Portanto, cada fase do projeto deve possuir uma atividade de verificação, validação e teste (VV&T) associada. Deficiências apontadas por estes procedimentos podem fazer com que, inclusive, se retorne em fases anteriores do projeto para que se façam as correções necessárias.

modelo deve ser finalizada podendo ser utilizada mais tarde quando houver a necessidade de se reutilizar o modelo para testar novos cenários.

11. **Implantação**. Normalmente, o objetivo final de qualquer projeto de simulação é o de promover um resultado concreto no sistema real. Quando o desempenho do sistema é aprimorado por uma nova política de operação ou configuração do sistema, determinada pelos experimentos da simulação, então, este objetivo foi atingido.

A princípio, um modelo de simulação pode ser implementado tanto em uma linguagem de programação (C, C++, Fortran, etc.) quanto em um pacote desenvolvido especificamente para este propósito, como o *Arena Simulation* (KELTON, 2001). No entanto, o tempo gasto na concepção, no desenvolvimento e na validação de um modelo implementado em uma linguagem de programação é muito superior em relação a um software específico de simulação. Estes dispõem de um grande conjunto de componentes matemáticos e estatísticos permitindo que o projetista atenha-se, exclusivamente, ao problema que está sendo modelado. Como exemplo, os pacotes possuem módulos para geração de sequências de números aleatórios que obedeçam a uma determinada distribuição de probabilidades. Por outro lado, as linguagens de programação, apesar de serem extremamente flexíveis, não possuem normalmente estes módulos, cabendo ao projetista a responsabilidade da sua implementação.

Ainda que os pacotes de simulação apresentem inúmeros módulos matemáticos e estatísticos, tornando mais rápido o processo de modelagem, muito ainda deve ser feito para que estes pacotes possibilitem um tempo menor no desenvolvimento de modelos e reduzam os pré-requisitos necessários aos profissionais que trabalham na área de simulação.

1.2 DEFINIÇÃO DO PROBLEMA

Dentre as inúmeras dificuldades na construção de modelos complexos de simulação, destaca-se a modelagem de processos estocásticos não estacionários. Sem ainda entrar no formalismo matemático da definição destes processos, considere-se, como um exemplo deste tipo de processo, uma central de atendimento ao público onde as chamadas ocorrem de forma aleatória, mas com taxas diferentes de chegadas por intervalos de 30 minutos. Esta aleatoriedade é simulada dentro dos softwares ou linguagens por meio da geração de números

aleatórios em observância a uma determina distribuição de probabilidades (Poisson, exponencial, normal, Weibull, entre outras). A quantidade de chamadas gerada no modelo, por período, deve convergir em média para o valor definido pela taxa de chegada. Frise-se, entretanto, que isto deve ocorrer apesar da aleatoriedade intrínseca às ocorrências destes chamados, dúvidas, reclamações, etc.

Outro problema, relacionado à quantidade de entidades geradas na simulação, é a possível existência de uma área ou espaço (buffer) demandada por estas entidades quando do seu tratamento pelo sistema. São inúmeros os exemplos, tanto na Administração quanto na própria área de Informática, da existência deste problema. Em ambientes de redes de computadores, caso cheguem mais mensagens (pacotes) em um roteador do que sua capacidade de tratamento, estas são armazenadas em uma área (buffer) do roteador até que seu limite seja atingido. Quando isto acontece, as mensagens subsequentes são descartadas, provocando a necessidade de reenvio pelo seu emissor.

Na área da Administração, mais especificamente em problemas logísticos, este descarte é absolutamente inaceitável ou impraticável. Portanto, o administrador deve fazer uma programação de envio de mercadorias de modo que o espaço disponível para seu armazenamento (silo, armazém, depósito) não seja completamente preenchido, o que causaria filas no seu entorno até que a descarga pudesse ser realizada. Além disso, outro requisito fundamental destes processos é que a quantidade de entidades geradas ao longo de um horizonte de planejamento deve convergir rigorosamente (e não em média, como nas distribuições de probabilidade) para um determinado valor. Considere-se um processo logístico no qual, ao final de um determinado período, uma quantidade exata de mercadoria deve ter sido transportada para um destino. Independentemente do meio de transporte utilizado (caminhões ou trens), ao final do horizonte de planejamento, a exata quantidade deve ter sido transportada para o destino.

Desta forma, a chegada destas mercadorias no destino se dá por meio de uma programação logística realizada pelos administradores, mas, ainda assim, dependente de outros fatores aleatórios que permeiam o seu envio, desde sua origem até seu destino. Então, além do volume necessário a ser atingido até um momento específico da simulação, a antecedência que deve ser adotada na geração destas entidades e a capacidade de armazenamento no destino, também precisam ser consideradas. A geração de entidades em

ambientes de simulação, habitualmente realizada por meio da amostragem aleatória simples (AAS) ou mesmo da amostragem descritiva (AD), proposta por Saliby (1989), não garante que, ao final da simulação, os totais necessários em cada destino convirjam rigorosamente (e não apenas em média) para os valores definidos.

A Figura 1.2 apresenta os componentes básicos do modelo de simulação do problema que se pretende resolver. Neste tipo de processo, a geração ou a entrada no modelo das entidades ocorre diretamente vinculada a um destino específico. Além disso, a quantidade de entidades geradas para um destino '*i*´ deve, ao final da duração de uma rodada da simulação, estar de acordo com os valores pré-estabelecidos ou definidos no início da simulação. Isto significa que a quantidade de entidades geradas por período da simulação (horas, dias, semanas, etc.) é um valor calculado deterministicamente. Entretanto, os intervalos de chegada continuam sendo aleatórios obedecendo a uma determinada distribuição não sendo, obrigatoriamente, um processo Poisson.

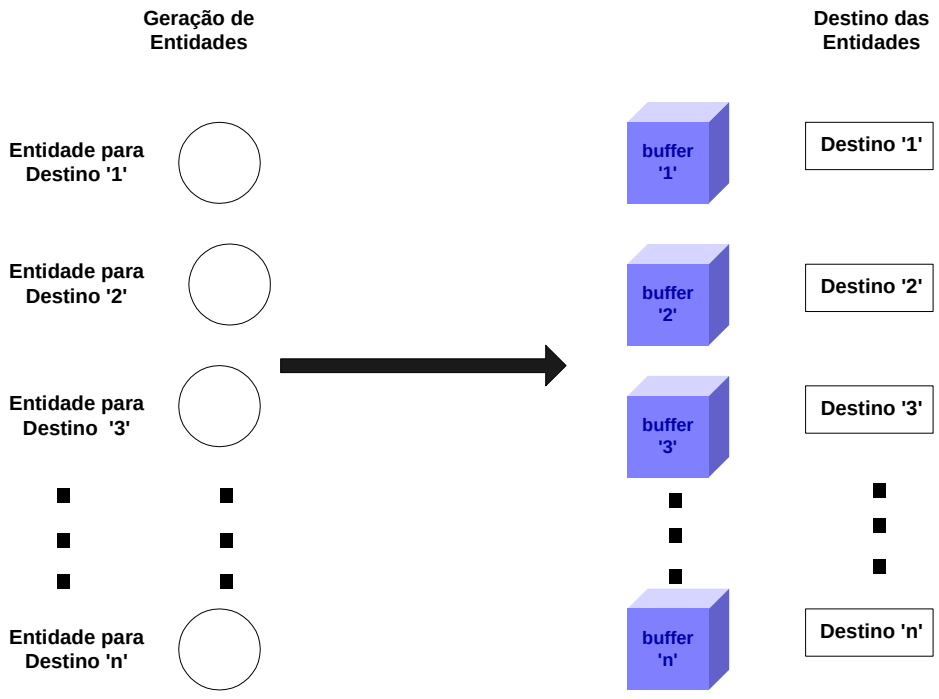

Figura 1.2 - Componentes básicos do modelo de simulação

Independentemente do mecanismo de transporte necessário para que as entidades cheguem aos seus respectivos destinos, as seguintes condicionantes devem ser levadas em consideração pelos métodos numéricos a serem desenvolvidos:

a) Até o instante previsto para a chegada de certa quantidade de entidades em um destino *´i´*, durante uma rodada da simulação, sua exata quantidade deve ter sido gerada pelo método, independentemente, do processo de geração das entidades ser estocástico;

b) Qual antecedência, em relação ao momento futuro previsto para as chegadas, deve ser adotada para o início da geração das entidades? Neste caso, quanto antes se iniciar o envio, maior é a probabilidade de que, no momento previsto e determinado, a quantidade exata já tenha chegado aos seus destinos correspondentes. Por outro lado, iniciar o envio com muita antecedência pode, simplesmente, causar o enchimento completo dos *buffers* ou, ainda, limitar a geração de entidades para outros períodos. Há um conflito entre segurança (garantir a chegada total das mercadorias na data prevista) e o espaço disponível para o seu armazenamento;

c) Ainda que os *buffers* não estejam completamente cheios, a quantidade de entidades, por unidade de tempo da simulação (que pode ser diária, semanal, mensal, etc.), geradas e enviadas aos destinos, não pode ser superior à capacidade de recepção e tratamento nos seus destinos, pois filas indesejáveis surgirão no entorno dos seus respectivos destinos.

Obviamente, o projeto de um modelo de simulação sujeito às condições acima não é um trabalho trivial. O uso das distribuições teóricas de probabilidades, usualmente disponíveis nos pacotes de simulação, é inadequado, pois não há garantia de que a condição apontada no item "*a"* seja atendida.

Outro problema para o projetista é que caberá a ele próprio decidir com qual antecedência iniciar a geração das entidades a fim de que, na data futura, a quantidade determinada já tenha sido alcançada. Desta forma, toda a lógica da complexidade desta decisão terá que ser modelada pelo projetista, causando, em consequência, um tempo elevado no desenvolvimento do projeto.

Por fim, também caberá ao projetista garantir a condição "*c*", o que igualmente não é trivial. Além da definição de qual distribuição de probabilidade usar, o projetista terá que definir seus parâmetros, garantindo que o valor médio desta distribuição esteja de acordo com

a capacidade de tratamento no destino. Mesmo assim, como se trata de uma distribuição de probabilidades, não há garantia de que a quantidade de entidades geradas convirja, naquele momento da simulação, para a real capacidade de tratamento nos seus destinos.

Um projetista de simulação, envolvido com a análise de sistemas reais que apresentem este grau de complexidade, depara-se com ambientes (softwares) de simulação, por mais sofisticados que sejam, que não dispõem de módulos ou ferramentas que o auxiliem na construção destes modelos. É necessário, por conseguinte, que o próprio projetista trate toda esta lógica de chegada de entidades, elevando em demasia o custo do projeto e, em virtude disto, fazendo com que os financiadores e interessados no projeto eventualmente desistam de sua implementação.

1.3 OBJETIVOS DA PESQUISA

Com o propósito de contornar as dificuldades descritas anteriormente, esta pesquisa tem como objetivo geral o desenvolvimento de métodos para geração de entidades, em ambientes de simulação, para processos estocásticos não estacionários, incorporando todas as relações matemáticas e estatísticas indispensáveis para a geração de entidades, liberando o projetista da simulação das complexidades expostas acima, quando do desenvolvimento de modelos para processos estocásticos não estacionários.

Ademais, esta pesquisa possui os seguintes objetivos específicos:

- Apresentar um referencial teórico na questão dos métodos numéricos envolvidos na pesquisa que permita a implantação dos métodos propostos em qualquer ambiente computacional, mesmo que seja uma linguagem de programação de uso geral;

- Identificação das variáveis de entrada e de decisão essenciais para a implementação dos métodos propostos. Estas variáveis, de certa forma, delimitarão o escopo ou alcance desses métodos;

- Exemplificar a aplicação dos métodos propostos em alguns modelos de simulação nas áreas de administração, redes de computadores e logística de transporte.

Fazendo uma analogia com a Figura 1.1, esta pesquisa tem por objetivo apresentar métodos para geração de entidades que auxiliem o projetista de simulação nas etapas 4 (coleta de dados) e 5 (implementação computacional do modelo). Notadamente, esta pesquisa concentra-se no desenvolvimento de métodos para geração de entidades, em ambientes de simulação, para processos estocásticos de renovação (*renewal stochastic processes*) não estacionários, cujas taxas por período são calculadas em virtude de uma demanda futura.

1.4 JUSTIFICATIVA DA PESQUISA

Levando em consideração que i) o uso da simulação deve mimetizar o mais fielmente possível o sistema real tendo em conta os objetivos definidos para o modelo e ii) a simulação destina-se à análise de sistemas complexos, então, é de fundamental importância que o projetista do modelo disponha de componentes de elevado nível de abstração, nos ambientes de desenvolvimento de simulação, para que possa criar com rapidez modelos confiáveis e precisos.

Brown (2000) apresenta as técnicas de desenvolvimento fundamentadas na Engenharia de Software Baseada em Componentes (ESBC). Este ramo da engenharia de software enfatiza a decomposição dos sistemas em **componentes** funcionais e lógicos, com interfaces bem definidas, propiciando significativa redução no tempo de desenvolvimento dos projetos e aumento da sua qualidade final. Vitharana (2003) identifica as principais vantagens da ESBC: i) redução no tempo do projeto em virtude de o desenvolvimento ser baseado em componentes já existentes, ii) redução nos custos de desenvolvimento dos componentes individuais, visto que são reutilizados em diversas aplicações, iii) aumento na qualidade dos componentes, em consequência de serem testados em diversas aplicações e iv) fácil substituição de componentes obsoletos por outros mais novos e sofisticados. Por outro lado, Crnkovic (2003) discute algumas dificuldades e desafios dentre os quais se destaca o conflito entre a **usabilidade** e a **reusabilidade** de um componente. Para que um componente seja amplamente reutilizável, deve ser suficientemente genérico e de fácil adaptação e ampliação para novas necessidades e, portanto, será um componente mais complexo, mais difícil de usar e vai requerer mais poder computacional para sua execução (mais caro para usá-lo).

Presman (2004) argumenta que os módulos (componentes de softwares) devem ser projetados de forma que as informações (procedimentos e dados) contidas dentro desses módulos sejam inacessíveis a outros que não necessitem dessas informações. O encapsulamento dos dados e procedimentos (*information hiding*) permite a definição de um conjunto de módulos independentes que se comunicam entre si por meio somente das informações necessárias. Desta forma, o desenvolvimento de sistemas modulares propicia grandes benefícios quando modificações são necessárias durante as fases de teste e de manutenção. Em virtude de que a maioria dos dados e funções está "escondida" do projetista, a probabilidade da introdução de erros é minimizada.

Balci (2001) apresenta uma metodologia para a certificação de componentes de softwares de simulação. Este autor reconhece que a fim de que se possa ter acesso aos benefícios desta tecnologia, há a necessidade do estabelecimento de um mercado para o desenvolvimento de aplicações em simulação baseada em componentes. Consequentemente, os seguintes benefícios econômicos podem ser alcançados: redução nos custos de projetos de modelagem e simulação, aumento da credibilidade dos projetos e a expansão e aplicabilidade de uma tecnologia menos custosa.

Kelton (2007) apresenta as dificuldades e limites na representação precisa da aleatoriedade dos sistemas reais usando as técnicas e métodos numéricos existentes. Conclui este autor que os softwares não provêem todo o suporte que poderia ser útil aos problemas apontados. As técnicas sugeridas pelo autor, como usar os dados reais disponíveis e o uso de distribuições empíricas (ao invés do processo de ajuste a uma distribuição específica), tampouco se aplicam ao problema proposto nesta pesquisa. Em um sistema no qual a taxa de chegada é determinada por uma necessidade ou demanda futura, o foco principal deve ser colocado no valor desta demanda futura (e de que forma ela será atendida nos períodos anteriores) e não nas chegadas individuais. Além disso, se fosse utilizada uma distribuição empírica para a geração de entidades, ainda assim, por mais fiel aos dados que seja esta representação, duas questões restariam não resolvidas em relação aos métodos propostos: a amostragem empírica está atrelada a uma situação de demanda específica e a geração de entidades por meio da distribuição empírica também não garante que a exata quantidade de entidades será gerada por período.

Kelton (2007. p. 39) reconhece que alguns softwares de simulação apresentam recursos para a representação de processos Poisson não estacionários, mas não possuem a capacidade de modelar processos aleatórios não estacionários mais genéricos. Este trabalho tem a intenção de preencher esta lacuna, ou seja, propor métodos numéricos que possam ser implantados em qualquer ambiente computacional e que possibilitem ao projetista do modelo concentrar-se, exclusivamente, na complexidade do problema quando se deparar com estruturas aleatórias não estacionárias que não sejam exclusivamente Poisson.

1.5 ESTRUTURA DO TRABALHO

Incluindo-se esta introdução, este trabalho foi desenvolvido numa sequência cuja finalidade é permitir ao leitor uma ideia clara desta pesquisa e de forma que a complexidade do assunto fosse apresentada de forma gradual, sem o comprometimento da compreensão do trabalho. Nesta introdução, colocou-se em evidência a necessidade do aprimoramento dos ambientes computacionais voltados à construção de modelos de simulação. Somente assim, as principais desvantagens da técnica de simulação (custos elevados associados à necessidade de profissionais altamente qualificados) poderão ser, de certa forma, mitigadas.

No capítulo 2 é apresentado o referencial teórico que embasa o desenvolvimento desta pesquisa. Este capítulo foi desenvolvido levando em consideração dois aspectos: i) a necessidade do entendimento teórico imprescindível para a condução da pesquisa e ii) a possibilidade da implantação dos métodos propostos em ambientes computacionais (linguagem de programação de uso geral) que não disponham de um arcabouço mínimo indispensável – ferramentas estatísticas - para a sua implantação. Por conseguinte, este capítulo tem a intenção de apresentar ao leitor os fundamentos estatísticos e métodos numéricos necessários para o uso dos métodos propostos em quaisquer ambientes computacionais. Neste referencial, são discutidos diversos aspectos de processos estocásticos, teoria de filas e simulação. No tocante à simulação, maior ênfase é dada aos aspectos centrais desta pesquisa: ajuste de um conjunto de dados a uma distribuição de probabilidades e geração de sequências aleatórias para processos estocásticos estacionários e não estacionários.

O desenvolvimento da pesquisa é feito no capítulo 3 em consonância com os objetivos estabelecidos para o trabalho. Considerando a complexidade e a natureza dos

processos estocásticos em foco, optou-se por subdividi-los em três categorias: i) processos com taxas constantes por período, de uma origem para múltiplos destinos, ii) processos com taxas constantes por período, de múltiplas origens para múltiplos destinos e iii) processos com taxas definidas por meio de planejamento.

O capítulo 4 tem por propósito validar os métodos propostos por meio de sua aplicação em sistemas reais em três áreas do conhecimento: administração (*call center*), redes de computadores e logística (recepção de mercadorias em ambientes portuários).

O capítulo 5 apresenta as considerações finais desta pesquisa, bem como as perspectivas de trabalhos futuros na área.

2 DISCUSSÃO TEÓRICA DOS MÉTODOS ENVOLVIDOS

Este capítulo apresenta e discute os principais métodos envolvidos neste projeto de pesquisa: processos estocásticos, teoria de filas, simulação, ajuste de um conjunto de dados a uma distribuição de probabilidades (*fitting*), geração de sequências aleatórias em ambientes de simulação e geração de sequências aleatórias para processos estocásticos não estacionários. Esta é a fundamentação teórica necessária tanto para a compreensão deste trabalho quanto para a aplicação dos métodos propostos em ambientes que eventualmente não disponham de ferramentas estatísticas essenciais.

2.1 PROCESSOS ESTOCÁSTICOS

Para a discussão a seguir, será assumido conhecimento prévio da teoria das probabilidades. Recomenda-se (MEYER, 1983) e (BARRY, 1996) para o estudo dessa teoria e das distribuições de probabilidades. A revisão conceitual de processos estocásticos baseou-se, principalmente em (HEYMAN; SOBEL, 2004) e (BEICHELT, 2006).

Um **processo estocástico** é definido como uma família de variáveis aleatórias $\{X(t) \mid t \in T\}$ definida num mesmo espaço de probabilidade $(\Omega, \mathcal{A}, \mathcal{P})$, onde Ω é o espaço amostral, $\mathcal{A}$ é a σ-álgebra (coleção de sub-conjuntos de Ω) e $\mathcal{P}$ é chamada de medida de probabilidade. Como uma variável aleatória é definida no espaço amostral Ω do experimento, então essa família de variáveis aleatórias é, na verdade, uma função de dois argumentos t e ω - $X(t,\omega)$ -, $t \in T$, $\omega \in \Omega$. Como pode ser observado na Figura 2.1, para cada valor de $t \in T$, tem-se uma variável aleatória $X(t,\omega)$ na forma de uma distribuição de probabilidades – $f_X(x)$. Para cada $\omega \in \Omega$ fixado, $X(t,\omega)$ é uma função de um único argumento (t) e é chamada de uma **realização** ou **trajetória** do processo estocástico. Quando se variam simultaneamente ω e t, tem-se uma família de variáveis aleatórias que constituem o processo estocástico.

Quando o espaço dos estados de um processo estocástico é discreto, diz-se que o processo é de **estado discreto** ou, alternativamente, **cadeia** (*chain*). Por outro lado, se o espaço dos estados é contínuo, então o processo é de **estado contínuo**. De maneira similar, se

o conjunto T é discreto, tem-se um processo estocástico de **parâmetro discreto**. Caso contrário, o processo é de **parâmetro contínuo**.

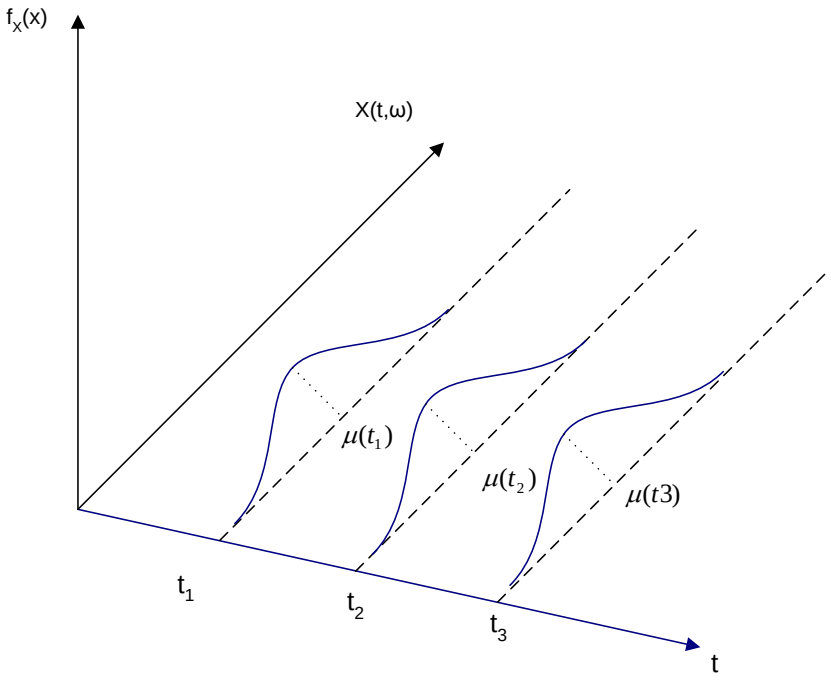

Figura 2.1- Um processo estocástico visto como uma família de variáveis aleatórias

Fonte: Adaptado de (MORETTIN; TOLOI, 2004)

A seguir serão dados exemplos de processos estocásticos, considerando um sistema de fila com clientes chegando para serem atendidos por um único atendente e, em seguida, partindo após terem sido atendidos.

Seja N_k a quantidade de clientes no sistema, no momento da partida do *k*-ésimo cliente (após ter sido atendido). Então, o processo estocástico $\{N_k \mid k = 1,2,...\}$ é um processo de parâmetro discreto e de estado discreto ($T = \{1,2,3....\}$ e $\Omega = \{0,1,2,...\}$).

Considere $X(t)$ a quantidade de clientes no sistema em um instante *t*. Desta forma, $\{X(t) \mid t \in T\}$ é um processo de parâmetro contínuo e de estado discreto ($T = \{t \mid 0 \leq t < \infty\}$ e $\Omega = \{0,1,2,...\}$).

Seja W_k o tempo de espera do *k*-ésimo cliente na fila até que seja atendido. Nesse caso, W_k é um processo de parâmetro discreto e de estado contínuo (T = {1,2,3....} e $\Omega = \{x \mid 0 \leq x < \infty\}$).

Por último, seja Y(*t*) o tempo de serviço cumulativo de todos os clientes atendidos no instante *t*. Nesse caso Y(*t*) é um processo de parâmetro e estado contínuo (T = $\{t \mid 0 \leq t < \infty\}$ e $\Omega = \{x \mid 0 \leq x < \infty\}$).

2.1.1 Classificação de processos estocásticos

Conforme pode ser visto na Figura 2.1, para um determinado instante $t = t_1$, $X(t_1)$ é uma variável aleatória que descreve o estado do processo no tempo t_1. Então, para um determinado valor x_1, $P(X(t_1) \leq x_1) = F_{X(t1)}(x_1)$. $F_{X(t1)}$ é a função cumulativa da variável aleatória $X(t_1)$ e, nesse caso, é chamada de **distribuição de primeira ordem** do processo estocástico. Por conseguinte, a função densidade de probabilidade (fdp) de $X(t_1)$ é dada por $f(X(t_1)) = \frac{\partial}{\partial X(t_1)} F_{X(t_1)}$. Considerando dois instantes t_1 e t_2, $X(t_1)$ e $X(t_2)$ são duas variáveis aleatórias no mesmo espaço de probabilidades. A distribuição conjunta destas duas variáveis aleatórias é conhecida como **distribuição de segunda ordem** do processo e é dada por $P(X(t_1) \leq x_1, X(t_2) \leq x_2) = F_{X(t1)\ X(t2)}(x_1, x_2)$. Expandindo o raciocínio, tem-se a **distribuição conjunta de n-ésima ordem** do processo X(t), $t \in T$, que é dada por:

$$F(x, t) = P(X(t_1) \leq x_1, \ldots, X(t_n) \leq x_n), \quad (2.1)$$

para todo $\mathbf{x} = (x_1, \ldots, x_n) \in \Re^n$ e $\mathbf{t} = (t_1, \ldots, t_n) \in T^n$ considerando que $(t_1 < t_2 \ldots < t_n)$.

Obter uma especificação completa de um processo estocástico, conforme pode ser visto pela expressão (2.1), não é uma tarefa trivial. Felizmente, para muitos processos, a função de distribuição conjunta de n-ésima ordem permanece a mesma sob translações no tempo. Isso nos conduz à definição de processo estritamente estacionário (*strictly stationary process*). Diz-se que um processo é **estritamente estacionário** se, para todo $n \geq 1$, sua função de distribuição conjunta de n-ésima ordem satisfaz a condição $F(x, t) = F(x, t+\tau)$, para todos os vetores $\mathbf{x} \in \Re^n$ e $\mathbf{t} \in T^n$ e todos os escalares τ de forma que $t_i+\tau \in T$. Seja $\mu(t) = E[X(t)]$ a média do processo e V(t) sua variância (ambas dependente do tempo), para

um processo estritamente estacionário $\mu(t) = \mu$ e $V(t) = \sigma^2$, para todo $t \in$ T, visto que todas as distribuições univariadas são invariantes no tempo. Consequentemente, para um processo estritamente estacionário, todos os seus demais momentos (assimetria e curtose, por exemplo) devem ser constantes.

Uma definição mais fraca de estacionariedade é usada normalmente em processamento de sinais e séries temporais. Um processo é dito ser **estacionário no sentido amplo** (*wide-sense stationarity*), **fracamente estacionário** (*weak-sense stationarity*), **estacionário de segunda ordem** (*second-order stationarity*) ou **estacionariedade de covariância** (*covariance stationarity*) se as seguintes condições forem atendidas:

- $\mu(t) = E[X(t)] = \mu$, isto é, a média do processo independe de *t*;
- $R(t_1, t_2) = R(0, t_2 - t_1) = R(\tau)$, isto é, a função de autocorrelação[2] (R) definida para um processo estocástico depende apenas do intervalo τ e não dos tempos t_2 e t_1;
- $R(0) = E[X^2(t)] < \infty$, isto é, o segundo momento deve ser finito.

Outra caracterização importante para um processo estocástico é feita restringindo-se a relação de dependência entre as variáveis aleatórias X(t). Se as variáveis aleatórias X(t) forem independentes, então, uma forma mais simples de (2.1) pode ser obtida:

$$\mathrm{F(x,\ t)} = \prod_{i=1}^{n} F(x_i; t_i) = \prod_{i=1}^{n} P[X(t_i) \le x_i] \tag{2.2}$$

---

[2] A função de autocorrelação de um processo estocástico é definida por $R(t_1, t_2) = E[X(t_1) * X(t_2)]$.

Se $t_1 = t_2$, então $R(t_1, t_1) = E[X^2(t_1)]$ e $Cov[X(t_1), X(t_2)] = R(t_1, t_2) - \mu(t_1)\mu(t_2)$. Observe que se o processo for estacionário no sentido amplo, então sua média é constante e não depende de *t*. Além disso, se a média do processo for igual a zero, então, a função de autocorrelação é igual a covariância.

Embora (2.2) represente uma significativa simplificação na análise dos processos estocásticos, assumir a completa independência das variáveis aleatórias de um processo estocástico nem sempre é razoável e alguma forma de dependência precisa ser considerada. A forma mais simples e comum de dependência é a de primeira ordem, também conhecida por dependência **markoviana.**

Um **processo markoviano** é um processo estocástico $\{X(t) \mid t \in T\}$, tal que $t_0 < t_1 < t_2 < \ldots t_n < t$, no qual a probabilidade condicional de $X(t)$, dados $X(t_0), X(t_1), \ldots, X(t_n)$, depende somente de $X(t_n)$. Posto de outra forma, a descrição do estado presente tem toda a informação que pode influir na evolução do próximo passo do processo, não sendo necessário o conhecimento dos estados anteriores do processo. Então, para um processo markoviano:

$$P[X(t) \leq x \mid X(t_n) = x_n, X(t_{n-1}) = x_{n-1}, \ldots, X(t_0) = x_0] = P[X(t) \leq x \mid X(t_n) = x_n] \qquad (2.3)$$

A definição (2.3) aplica-se tanto para processos de parâmetro discreto quanto contínuo. Quando o processo markoviano é de parâmetro discreto chama-se, então, **cadeia de Markov** (*Markov chain*).

2.1.2 Processo de Renovação (Renewal Process)

Levando em consideração a equação (2.2), um **processo de renovação** (PR) (*renewal process*) é definido como um processo estocástico de tempo discreto $\{X_n \mid n = 1,2, \ldots\}$ no qual $X_1, X_2, \ldots$ são variáveis aleatórias não negativas, independente e identicamente distribuídas. O tempo entre falhas sucessivas de um equipamento pode ser representado pelas variáveis aleatórias independentes e identicamente distribuídas $\{X_n \mid n = 1, 2, \ldots\}$, que são as variáveis aleatórias de um processo de renovação. Quando todas as variáveis $\{X_n \mid n = 1,2, \ldots\}$ são independentes e identicamente distribuídas chama-se de **processo de renovação ordinário** (*ordinary renewal process*) (BEICHELT, 2006, p. 155).

Em algumas situações, não é razoável assumir que todas as variáveis tenham a mesma distribuição. Considere a avaliação de um processo estocástico de substituição de um

equipamento após sua falha, mas que no instante $t = 0$ o equipamento não seja novo e que se esteja interessado na contagem dessas substituições ao longo do tempo. Nesse caso, tem-se um **processo de renovação generalizado** (*delayed ou generalized renewal process* - DRP).

Definição formal – DRP: (HEYMAN; SOBEL, 2004, p. 137), Sejam X_1, X_2,**...** variáveis aleatórias independentes com $G(x)=P(X_1 \leq x)$ e $F(x)=P(Xi \leq x)$ para $i = 2,3,$ **...**, sendo $G(x) \neq F(x)$. Então, X_1, X_2,**...** é um processo de renovação generalizado. Assim sendo, a distribuição da variável aleatória X_1 é diferente das demais variáveis X_i´s.

2.1.3 Processo Bernoulli e Processo Binomial

Um **processo Bernoulli** é um processo estocástico de tempo e estado discreto, estacionário no sentido estrito, cuja sequência de variáveis aleatórias independentes X_1, X_2, **...** , X_n representam ensaios de Bernoulli (*Bernoulli trials*). Portanto, cada variável aleatória X_i assume o valor 1 (sucesso) ou 0 (falha). Além disso, P[X_i=1]=p, P[X_i=0]=1-p, E[X_i]=p, E[X_i^2]=p e VAR[X_i]=p(1-p). Como exemplo deste tipo de processo, considere um canal de um sistema de telecomunicações que transmita uma sequência de *bits* cuja probabilidade de transmissão correta é *p* e incorreta (1-*p*).

Suponha agora outro processo estocástico $\{Y_n \mid n = 1, 2, \ldots\}$, sendo $Y_n = X_1+X_2+\ldots+X_n$. Y_n é a soma parcial de *n* variáveis aleatórias de um processo de Bernoulli. Então, Y_n é um processo de estado e tempo discreto conhecido por **processo binomial**. Desta forma, $P[Y_n = k] = \binom{n}{k} p^k (1-p)^{n-k}$, $E[Y_n] = np$ e $VAR[Y_n] = np(1-p)$. Prosseguindo no exemplo do sistema de telecomunicações, suponha que agora se esteja interessado na contagem do número de bits transmitidos corretamente numa determinada sequência. Esta contagem é um processo estocástico binomial.

Várias generalizações podem ser feitas no processo Bernoulli. Uma delas é considerar que cada variável X_i tenha um parâmetro *p* diferente para cada ensaio. Neste caso, tem-se um **processo Bernoulli não-homogêneo**. Outra generalização possível é permitir que

cada ensaio possa produzir mais que dois resultados. Assim, $\{X_i \mid i = 1, 2, 3, \ldots\}$é o processo estocástico e $\sum_{i=1}^{N} p_i = 1$, no qual ***N*** representa a quantidade possível de resultados para cada ensaio. Para esse caso, o processo estocástico $\{Y_n \mid n = 1, 2, \ldots\}$, sendo $Y_n = X_1 + X_2 + \ldots + X_n$, é uma cadeia de Markov conhecida por **passeio aleatório** (*random walk*).

Por fim, respeitando o escopo desta pesquisa, outra generalização possível é estudar o comportamento da transformação do processo binomial de tempo discreto para tempo contínuo. Da teoria das probabilidades, sabe-se que a distribuição de Poisson pode ser obtida da distribuição binomial. Assim sendo, como a soma de um processo Bernoulli é um processo binomial, cuja função *fdp* é igual a *b(n;k;p)*, se *n* é grande e *p* suficientemente pequeno, pode ser provado que *b(n;k;p)* aproxima-se da *fdp* de uma distribuição Poisson *f(k;np*), na qual np = λt, apresentada em 2.1.4.

2.1.4 Processo Poisson

Um **processo Poisson** é um PR de tempo contínuo e estado discreto. Considere-se o processo $\{N(t) \mid t > 0\}$ sendo *N(t)* a quantidade de eventos que ocorreram no intervalo (0,*t*]. Os eventos ocorrem sucessivamente e os intervalos entre eventos são independentes e distribuídos de acordo com uma distribuição exponencial ($F(t) = 1 - e^{-\lambda t}$). Sabe-se que a esperança de uma variável aleatória que siga uma distribuição exponencial é $E(X) = 1/\lambda$. Portanto, λ representa a taxa média do processo Poisson.

Quando a taxa λ não varia de acordo com o tempo, tem-se um **processo Poisson homogêneo** e a probabilidade de que haja a ocorrência de *k* eventos no intervalo (0, *t*] é dada por $P[N(t) = k] = \frac{(\lambda t)^n * e^{-\lambda t}}{k!}$, para $k>0$. Uma generalização importante desse tipo de processo é quando se considera que λ é uma função do tempo. Nesse caso, tem-se um

processo Poisson não-homogêneo[3] e o número de chegadas é distribuído de acordo com $m(t) = \int_0^t \lambda(x)dx$.

O processo Poisson é extremamente importante para a teoria de filas e para a área de confiabilidade. Como exemplos em sistemas reais pode-se citar o número de chamadas que ocorrem em um *call-center*, a quantidade de requisições de arquivos em um servidor de rede (ambientes computacionais), a quantidade de requisições de páginas em um servidor *Web* e a quantidade de falhas que ocorrem em um grande número de componentes de um sistema que esteja operando inicialmente sem nenhum problema.

A **teoria de filas** é uma abordagem analítica para a análise de sistemas de filas (lembrar que um sistema de filas é um processo estocástico). Normalmente, os clientes chegam aleatoriamente e, em geral, o tempo de serviço que demandam também não é constante.

2.1.5 Teoria de Filas (aplicações e limitações)

De acordo com Cooper (1982), o termo *teoria de filas* é frequentemente usado para denominar a teoria matemática que descreve o comportamento de linhas de espera ou filas. (ERLANG, 1909), em seu artigo seminal de teoria de filas, estabelece as bases da importância da distribuição de *Poisson* e, em consequência, da distribuição *exponencial* para a teoria do congestionamento.

O fenômeno das filas (PRADO, 2004) ocorre a cada instante no cotidiano das pessoas e no mundo dos negócios, sendo fundamental, portanto, que este problema seja tratado cientificamente. Assim sendo, a finalidade da teoria de filas é estabelecer

[3] Muitos autores usam os termos homogêneo/estacionário e não-homogêneo/não-estacionário como sinônimos quando se referem aos processos Poisson. Trivedi (2002, p. 307) sustenta que, considerando que a esperança de um processo Poisson é λt (portanto dependente do tempo), o processo Poisson não é, por definição, estacionário.

numericamente qual é o valor destas incertezas, como por exemplo, quanto tempo, em média, os clientes permanecerão na fila ou quantos clientes estarão, em média, na fila. Kendall (1953) introduziu a notação amplamente utilizada para a representação de sistemas de filas. Mais tarde, essa notação foi modificada por Lee (1966) para a atualmente conhecida como notação de Kendall-Lee.

A teoria de filas estabelece resultados analíticos para as variáveis de interesse destes sistemas. Então, é lícito indagar qual seria a necessidade da simulação para a análise de sistemas com filas, se a própria teoria de filas já estabelece estes resultados analíticos. Tanto Cooper (1982) quanto Kelton, Sadowski & Sadowski (2001) concordam que os modelos da teoria de filas são relativamente simples e, em consequencia, adequados para modelar sistemas reais simples ou, apenas, partes de sistemas reais mais complexos. A teoria de filas é frequentemente muito restritiva do ponto de vista matemático para que possa modelar fielmente as diversas situações do mundo real. Estas restrições surgem das premissas assumidas na teoria de filas, que nem sempre são adequadas ao mundo real. Por exemplo, no modelo *M/M/1,* os intervalos de chegadas obedecem a uma distribuição exponencial e, assim, não há limite superior para o intervalo entre chegadas. Entretanto, este limite pode existir no sistema real que está sendo modelado e, em virtude disto, os resultados podem ser diferentes dos apontados analiticamente.

(GONZALES; BOTTER, 2002) é outro trabalho bastante ilustrativo que coteja a aplicação da teoria de filas com a simulação em ambientes portuários. Para estes autores, embora o custo de um projeto de simulação seja mais elevado, a aplicação da simulação permite um maior detalhamento dos processos portuários, de modo a permitir a avaliação, de forma mais consistente, das esperas dos navios, contribuindo para um cálculo mais realista da sobreestadia (*Demurrage)* a ser paga ao armador pelo afretador ou seu consignatário.

2.2 SIMULAÇÃO

O mesmo sistema de fila *M/M/1* ou outros quaisquer podem ser modelados por meio de uma linguagem ou software de simulação. Para fins de exemplificação, neste trabalho será usado o *Arena Simulation* (KELTON, 2001; FREITAS FILHO, 2001; ROCKWELL SOFTWARE, 2005).

Os resultados analíticos da teoria de filas são válidos para um sistema em equilíbrio ou em estado estacionário. Quando se trabalha com simulação, inicialmente é preciso definir por quanto tempo a simulação vai rodar (um dia, uma semana, um mês ou apenas algumas horas). Normalmente, esta definição é feita de acordo com a própria natureza do sistema que está sendo modelado. Outra definição importante é por quanto tempo a simulação será executada até que o estado estacionário seja atingido; este tempo é conhecido por *"Warm-Up"*. Para o cálculo estatístico das variáveis de interesse, este período deve ser expurgado a fim de que não provoque uma distorção nos resultados.

Por fim, outro parâmetro extremamente importante é quantas replicações/rodadas ou amostras da simulação serão feitas. De nada adianta simular o comportamento de uma fila apenas um dia. Como na simulação as variáveis aleatórias são fornecidas usando-se distribuições de probabilidades, executar a simulação por apenas um dia não significa que naquele dia teremos um dia "típico". Em termos estatísticos, o que se teria é uma amostra de tamanho 1, o que, indubitavelmente, é muito pouco para se tirar conclusões de como funciona o sistema que está sendo modelado.

A fim de se estabelecer uma comparação com a teoria de filas, o Quadro 2.1 mostra os valores obtidos por meio da simulação e os valores analíticos da teoria de filas para um sistema *M/M/1*. Foi definido que a simulação é executada durante 600 horas com um período de aquecimento de 30 horas e foram executadas 30 rodadas.

Ao final da simulação, o *Arena* apresenta diversos relatórios com os dados estatísticos das variáveis de interesse. Ressalte-se que, como os valores obtidos pela simulação são estatísticos, o *Arena* calcula também o Intervalo de Confiança (IC) em 95% para os valores médios calculados.

Variáveis	Teoria de Filas (Valor médio)	Simulação	
		Valor Médio	Intervalo de Confiança
L - Número médio clientes no sistema	3	3,031	0.18
Lq - Número médio clientes na fila	2.25	2,277	0.17
W - Tempo médio de um cliente no sistema	1	1,003	0.05
Wq - Tempo médio de um cliente na fila	0.75	0,7531	0.05

Quadro 2.1- Resultados analíticos e por simulação para o sistema M/M/1, com λ=3 e μ=4

O IC para uma determinada medida de desempenho que está sendo aferida pela simulação é um componente fundamental para se analisar os resultados. O IC é um intervalo numérico que tem a probabilidade 1-α de incluir o valor verdadeiro da medida de desempenho, onde 1-α é o nível de confiança do intervalo. Suponha que sejam executadas 5 rodadas de um modelo *M/M/1*. O Quadro 2.2 apresenta os resultados da variável W_q para as 5 rodadas. Considerando os dados do Quadro 2.2, pode-se calcular a média amostral $\overline{X} = \sum_{i=1}^{n} \frac{X_i}{n} = 0.8463$ e a desvio padrão amostral, $S = \sqrt{\frac{\sum_{i=1}^{n} X_i^2 - n * \overline{X}^2}{n-1}} = 0.1973$. O IC é calculado pela expressão $(\overline{X} - h, \overline{X} + h)$, onde $h = t_{n-1,1-\alpha/2} * \frac{S}{\sqrt{n}}$. O valor de h é denominado por meia largura (*half width* – HW) do IC. O valor $t_{n-1,1-\alpha/2}$ é a quantidade que limita a região crítica da distribuição *t de student*. Para o caso em questão, tem-se que $t_{4,.975} = 2.78$. Então, $h = 2.78 * \frac{0.1973}{\sqrt{5}} = 0.25$ e o IC $(0.60,\ 1.09)$. Isto significa que, ao executar-se cinco rodadas, a probabilidade é de que em 95% destas rodadas o valor médio do tempo de espera em fila esteja entre 0.60 e 1.09.

Replicação Número, i	W_q, X_i
1	1.1832
2	0.7555
3	0.8047
4	0.8192
5	0.6688

Quadro 2.2- Resultados para o tempo de espera na fila para 5 rodadas ou replicações

As seguintes relações são verdadeiras (BANKS, 1998):

1) Aumentando-se o número de replicações, o comprimento do IC diminui;

2) À medida que o nível de confiança aumenta (para o mesmo número de replicações), o comprimento do IC também aumenta;

3) À medida que a variância *S* aumenta, o comprimento do IC aumenta.

Law & Kelton (2000, p. 251) afirmam que, de acordo com a própria experiência, dados de saída de uma simulação são quase sempre correlacionados. Pode-se provar que o estimador da variância S^2 é não-viesado desde que os dados sejam independentes e, portanto, não-correlacionados. Por conseguinte, deve-se tomar cuidado ao se calcular o IC usando o estimador S^2da variância. Considere um sistema de fila *M/M/1* com intervalos de chegada IID (intervalos mutuamente independentes e com mesma distribuição de probabilidade) A_1, A_2, A_3,... e tempos de serviço, também IID, S_1, S_2 ,S_3, ... Seja o processo estocástico D_1, D_2, D_3, ... no qual D_i é o tempo de espera na fila para o *i*-ésimo cliente. Neste caso pode-se mostrar que D_i e D_{i+1} são positivamente correlacionados e, portanto, não é correto usar o estimador S^2 da variância para se calcular o IC. É fundamental, neste caso, executar *n* rodadas extraindo-se, para cada rodada, o valor médio dos tempos de espera. Como as rodadas são independentes e não-correlacionadas, o estimador S^2 pode ser usado para calcular o IC das médias de cada rodada.

2.2.1 Modelagem de Processos Poisson não estacionários no Arena

Considerando que o objetivo do trabalho é o desenvolvimento de modelos para processos não estacionários, a seguir mostra-se o ferramental disponível no *Arena* para esta finalidade.

O *Arena* dispõe de um módulo chamado *Schedule* que permite a modelagem de processos *Poisson* não estacionários. O usuário define as taxas de chegadas por período por meio do módulo *Schedule* e usando um módulo *Create* estabelece que as entidades entrarão no modelo de acordo com as taxas previamente definidas no *Schedule*. Automaticamente, o *Arena* gera as entidades com intervalo entre chegadas distribuídos exponencialmente, garantindo os valores médios das taxas entre as rodadas da simulação.

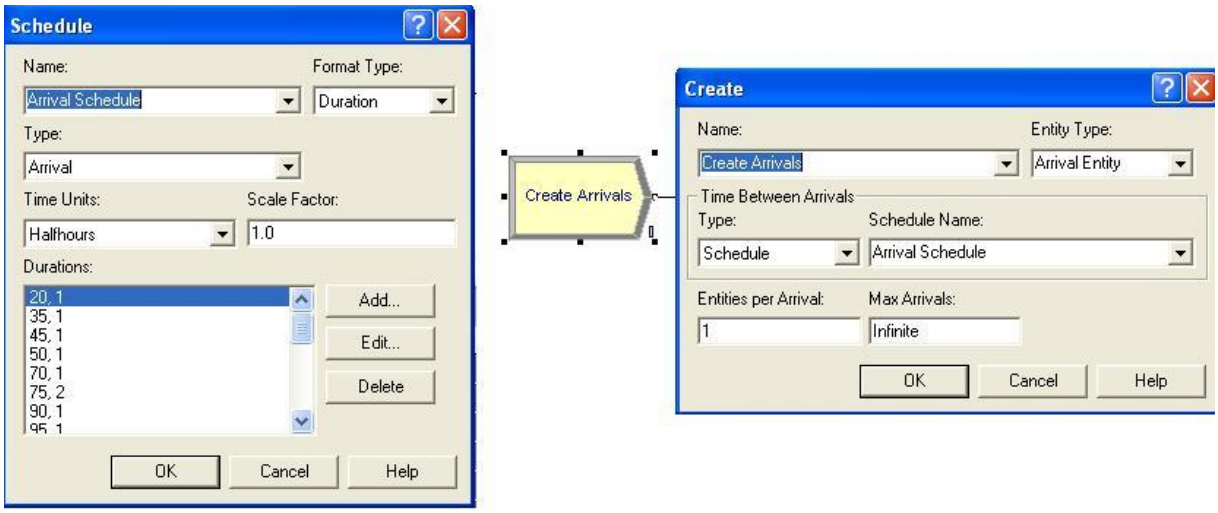

Figura 2.2- Ferramenta '*Schedule*' disponível no *Arena*

2.2.2 Deficiências de Modelagem para Processos não estacionários no Arena

Duas grandes limitações decorrem desta forma de implementação:

- Não há suporte automático para processos que não sejam Poisson (outras distribuições para os intervalos entre chegadas que não a exponencial);
- A solução não apresenta escalabilidade - habilidade que um sistema possui de se acomodar quando a dimensão do problema aumenta – (DUBOC; ROSENBLUM; WICKS, 2006).

Considere-se uma modelagem de uma rede de computadores com 100 nós (roteadores) e que se esteja interessado na análise do tráfego de dados entre estes nós. Neste caso, seria necessária a geração de pacotes (mensagens) entre todos os nós, o que daria 10^4 definições de *Schedule* e a mesma necessidade da inserção de módulos *Create* dentro do modelo. Além de se tornar inviável a construção do modelo à medida que sua dimensão aumenta, resta ainda não resolvida, no *Arena*, a questão dos intervalos entre chegadas, quando esses forem diferentes da distribuição exponencial.

Tanto para a simulação quanto para a teoria de filas, é de fundamental importância a definição, na modelagem de um sistema real (cujo comportamento seja estocástico e estacionário), de qual é a distribuição de probabilidade (e seus parâmetros) que melhor

representa um determinado conjunto de dados. Por outro lado, para a simulação, cabe ao software utilizado ou ao próprio projetista do modelo (quando se utiliza uma linguagem de programação de uso geral) gerar uma sequência numérica e aleatória que atenda àquela distribuição de probabilidade com seus parâmetros especificados. Estes tópicos, ajuste de um conjunto de dados a uma distribuição de probabilidades e geração de sequências aleatórias em ambientes de simulação, serão abordados a seguir. Esses assuntos são de extrema relevância para a compreensão e o desenvolvimento desta pesquisa, visto que o seu objetivo é desenvolver métodos para geração de entidades, em ambientes de simulação, para processos estocásticos não estacionários.

2.3 AJUSTE DE UM CONJUNTO DE DADOS A UMA DISTRIBUIÇÃO DE PROBABILIDADES

O ajuste (*fitting*) de um conjunto de dados (que representam o comportamento de uma variável aleatória) a uma distribuição de probabilidades consiste em encontrar uma função matemática que represente, o mais fielmente possível, o comportamento dessa variável aleatória. Um projetista de simulação, frequentemente, enfrenta esse problema em diversos pontos da construção do modelo. Por exemplo, o projetista possui diversas amostras (x_1, x_2,..., x_n) relativas aos intervalos de chegadas de clientes no sistema. O que se deseja saber é se estas amostras pertencem a uma função densidade de probabilidade (fdp) $f(x,\theta)$ onde θ é um vetor de parâmetros que precisam ser estimados com os dados disponíveis das amostras.

Grosso modo, o ajuste a uma distribuição de probabilidade é feito com os seguintes passos (RICCI, 2005):

1) Selecionar a fdp que melhor modele os dados;

2) Estimar os parâmetros da fdp;

3) Avaliar a qualidade da estimação aplicando testes estatísticos que meçam a capacidade de ajustamento (*Goodness of fit*).

A seguir estes passos serão mais bem detalhados.

2.3.1 Selecionar a fdp que Melhor Modele os Dados

O primeiro passo para o ajuste dos dados é escolher uma função matemática (fdp) que melhor represente esse conjunto de dados. Como uma primeira abordagem, pode-se usar histogramas ou outras técnicas gráficas.

Um histograma (Figura 2.3) já pode mostrar o comportamento de assimetria (*skewness*) e das caudas, bem como a presença ou não de comportamento multimodal no conjunto de dados. Além disso, o histograma pode ser comparado com a forma gráfica das funções de distribuições teóricas. Os gráficos a seguir referem-se a uma amostra com 1000 valores de uma variável aleatória exponencial X~EXP(1).

Um histograma mostra uma estimativa do gráfico da função densidade correspondente à distribuição dos dados X_1, X_2, **…** , X_n. A construção de um histograma é feita separando-se o conjunto de dados em *k* intervalos disjuntos e adjacentes [b_0, b_1), [b_1, b_2), **…** , [b_{k-1}, b_k), sendo que todos os intervalos possuem o mesmo tamanho $\Delta b = b_j - b_{j-1}$. Então, para j = 1, 2, **…**, *k*, h_j é a proporção dos dados X_i´s que estão contidos no j-ésimo intervalo [b_{j-1}, b_j). Assim, define-se *h(x)* que é uma função de *x*.

$$h(x) = \begin{cases} 0 \ se \ x < b_0 \\ h_j \ se \ b_{j-1} \le x < b_j \ \ para \ \ j = 1, 2, ..., k \\ 0 \ \ se \ b_k < x \end{cases}$$

Law & Kelton (2000, p. 336) mostram que o formato da função h(x) assemelha-se ao da *fdp f* dos dados, visto que: $P(b_{j-1} \le X \le b_j) = \int_{b_{j-1}}^{b_j} f(x)dx = \Delta b f(y)$ para qualquer valor $y \in (b_{j-1}, b_j)$. A primeira equação decorre diretamente da definição de uma variável aleatória contínua e a segunda decorre do teorema do valor médio do cálculo. O grande problema na utilização de histogramas é determinar qual é o valor ideal de Δb ou de forma análoga o valor de *k*.

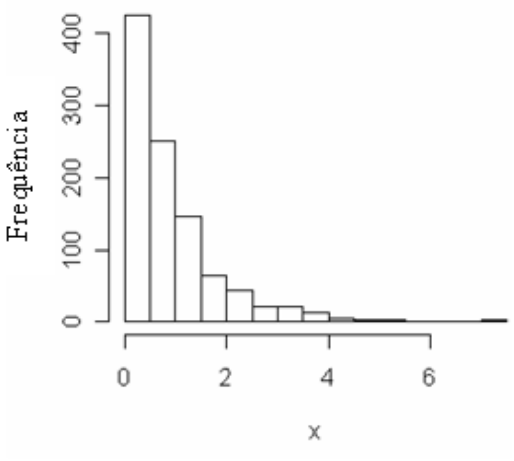

Figura 2.3 - Histograma de uma variável X~EXP(1)

Outro gráfico que pode ser usado é o gráfico da função densidade dos dados, conforme mostra a Figura 2.4.

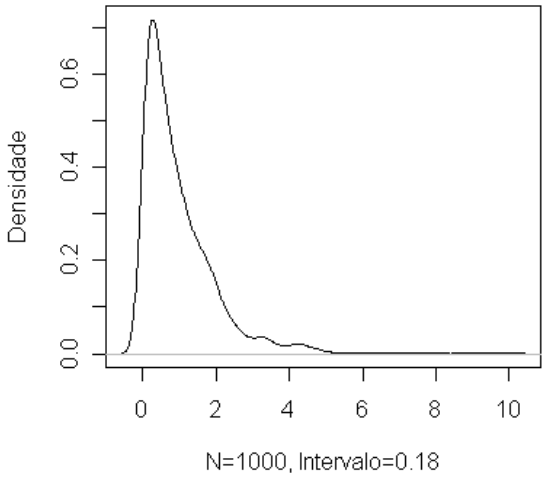

Figura 2.4 - Gráfico da função densidade dos dados de uma variável X~EXP(1)

Além disso, um gráfico (Figura 2.5) que mostre a função cumulativa dos dados pode ajudar a compará-la com a função cumulativa de uma distribuição teórica.

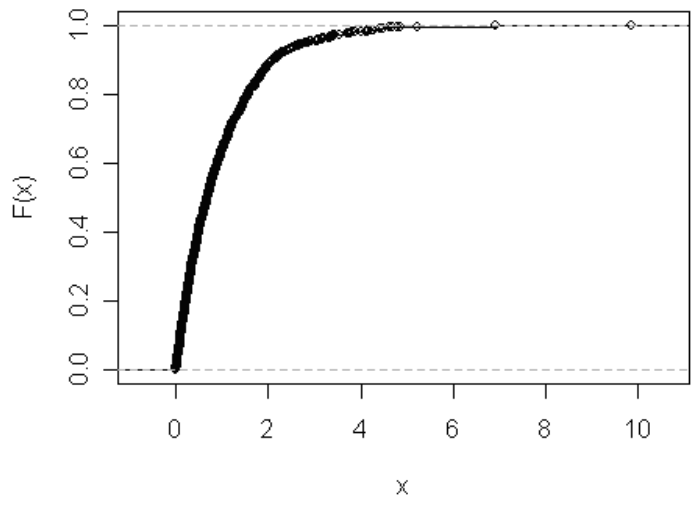

Figura 2.5 - Gráfico da função cumulativa da amostra

Por último, uma forma mais eficaz para se comparar distribuições empíricas e teóricas é comparar os *quantis* das distribuições por meio de um gráfico *Quantis-Quantis* (Q-Q). O gráfico Q-Q é um gráfico dos dados ordenados da amostra contra os *quantis* esperados de uma determinada distribuição que se deseja comparar. Quanto mais próximos os pontos estiverem da bissetriz do primeiro quadrante, mais próximos estarão os dados da distribuição que se considera, conforme mostra a Figura 2.6. Quanto mais os dados divergirem da linha de referência, maior é a evidência de que o conjunto de dados provém de uma distribuição diferente.

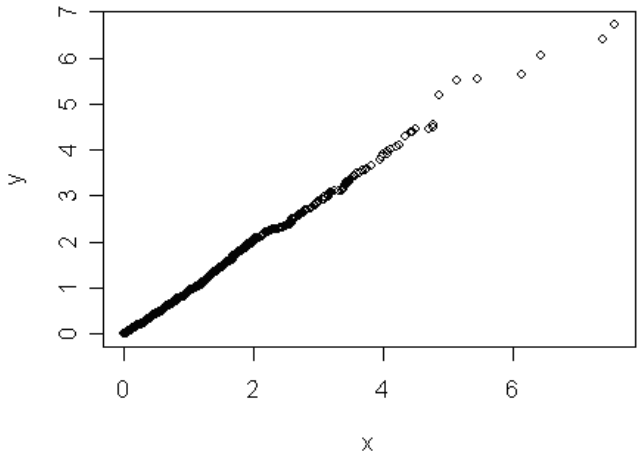

Figura 2.6 - Gráfico Q-Q comparando a amostra X com uma distribuição EXP(1)

2.3.2 Estimar os Parâmetros da fdp

Após ter sido escolhida a distribuição que melhor modela a amostra, é necessário estimar seus parâmetros. Inicialmente, uma abordagem bastante simplista, no caso da distribuição normal, é estimar os parâmetros da distribuição usando os dados da amostra. Por exemplo, a média da distribuição é estimada calculando a média da própria amostra. Este método é conhecido por **método analógico**.

Outro método bastante conhecido e utilizado é o **método dos momentos** *(*CAMERON, A. C.; TRIVEDI, P. K., 1998). A estimação dos parâmetros é feita igualando os momentos da amostra aos momentos da população e, então, resolvem-se estas equações a fim de que os parâmetros sejam estimados. A título de exemplo, será mostrado como se estimam os parâmetros de uma distribuição Beta. Supondo que X seja uma variável aleatória IID, de acordo com uma distribuição *Beta*, então a fdp de X é data pela expressão:

$$f(x;\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} \tag{2.4}$$

O primeiro momento de X (a esperança de X) e o segundo momento (a variância de X) são dados pelas expressões:

$$E(X) = \frac{\alpha}{\alpha+\beta} \tag{2.5}$$

$$\text{Var(X)} = = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \tag{2.6}$$

O primeiro e o segundo momentos da amostra podem ser calculados por:

$$m_1 = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.7}$$

$$m_2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - m_1)^2 \tag{2.8}$$

Igualando os momentos populacionais (2.5) e (2.6) aos momentos amostrais (2.7) e (2.8), tem-se que:

$$\alpha = m_1\left(\frac{m_1(1-m_1)}{m_2}-1\right) \tag{2.9}$$

$$\beta = (1-m_1)\left(\frac{m_1(1-m_1)}{m_2}-1\right) \tag{2.10}$$

Em algumas situações, conforme mostram Campos, Nascimento, e Studart (2003), o método dos momentos foi superado pelo **método da máxima verossimilhança** (*maximum likelihood*), em virtude de este apresentar uma probabilidade maior de que os estimadores estejam mais próximos dos valores a serem estimados. Entretanto, em outros casos, como o da distribuição *Gama*, as equações de verossimilhança são extremamente complexas para serem resolvidas sem o uso computacional, enquanto que no método dos momentos os estimadores podem ser facilmente calculados. O método dos momentos pode ser usado como uma primeira aproximação para as equações de máxima verossimilhança, obtendo-se, em seguida, melhores aproximações.

Supondo uma variável aleatória com uma fdp *f(x,θ),* pode-se estimar o vetor dos parâmetros desconhecidos (θ) de acordo com os valores da amostra (x_1, x_2,..., x_n). O método da máxima verossimilhança consiste em definir uma função matemática conhecida como função de verossimilhança da amostra. Grosso modo, a verossimilhança de um conjunto de dados é a probabilidade de se obter aquele conjunto de dados, dada a fdp escolhida. Essa expressão matemática contém os parâmetros desconhecidos e os parâmetros que maximizam a verossimilhança da amostra são conhecidos por **estimativas da verossimilhança da amostra** (*maximum likelihood estimates* – MLE). A função de verossimilhança é definida por:

$$L(x_1, x_2, ..., x_n, \theta) = \prod_{i=1}^{n} f(x_i, \theta) \tag{2.11}$$

Assim sendo, a estimativa MLE consiste em encontrar θ que maximize $L(x_1, x_2, ..., x_n, \theta)$ ou sua função logarítmica. Pode-se, então, empregar métodos matemáticos de análise (derivadas parciais, por exemplo), quando a função de verossimilhança for simples,

ou métodos iterativos. Para o caso da distribuição *gamma*, cuja fdp é dada por $f(x;\alpha,\lambda)=\frac{\lambda^{\alpha}}{\Gamma(\alpha)}x^{\alpha-1}e^{-\lambda x}$ para x>0, a função de verossimilhança é dada por:

$$L(\mathrm{x}_1,\mathrm{x}_2,...,\mathrm{x}_n,\theta)=\prod_{i=1}^{n}\mathrm{f}(\mathrm{x}_i,\alpha,\lambda)=\prod_{i=1}^{n}\left(\frac{\lambda^{\alpha}}{\Gamma(\alpha)}\right)x_i^{\alpha-1}e^{-\lambda x_i}$$

$$=\left(\frac{\lambda^{\alpha}}{\Gamma(\alpha)}\right)^{n}\left(\prod_{i=1}^{n}x_i\right)^{\alpha-1}e^{-\lambda\sum_{i=1}^{n}x_i}$$

$$\log(L)=n\alpha\log(\lambda)-n\log(\Gamma(\alpha))+(\alpha-1)\sum_{i=1}^{n}\log(x_i)-\lambda\sum_{i=1}^{n}x_i \qquad (2.12)$$

2.3.3 Avaliar a Qualidade da Estimação Aplicando Testes Estatísticos que Meçam a Capacidade de Ajustamento (*Goodness of fit*)

A qualidade do ajuste de um modelo a um conjunto de dados pode ser avaliada basicamente de duas formas: i) por meio de métodos gráficos que comparem visualmente as semelhanças e diferenças entre o modelo e os dados observados e ii) por meio de medidas que mostrem numericamente a precisão do modelo.

Como exemplo da primeira abordagem, pode-se exibir um gráfico que mostre conjuntamente a curva da fdp ajustada com o histograma dos dados, conforme mostra a Figura 2.7.

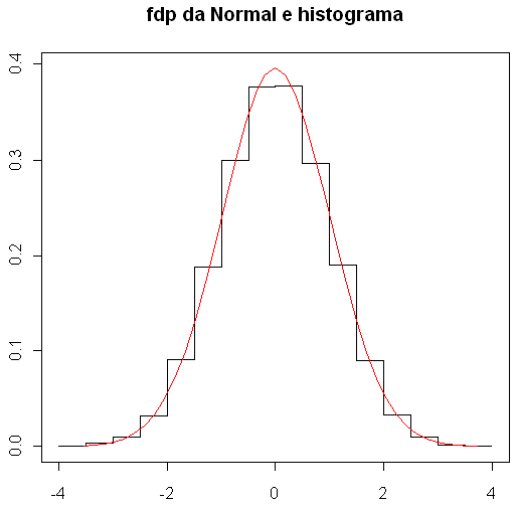

Figura 2.7 - Histograma dos dados com a curva da fdp estimada

Entretanto, para que se possa assumir, com maior precisão, que o conjunto de dados em análise provém do modelo estabelecido, é necessário um método mais rigoroso. Apesar de existirem diversas técnicas estatísticas para medir a qualidade do ajuste, neste referencial teórico apresentam-se as técnicas Qui-Quadrado (*Chi-Square)* (SNEDECOR; COCHRAN, 1989) e *Kolmogorov-Smirnov* (CHAKRAVARTI; LAHA; ROY, 1967), que são amplamente utilizadas em diversos pacotes de simulação - inclusive no *Arena Simulation*.

2.3.3.1 Teste de Ajuste Qui-Quadrado

O teste de ajuste qui-quadrado indica se os dados da amostra provêm de uma distribuição específica. Este teste é uma forma de teste de hipóteses, onde a hipótese nula e a hipótese alternativa são:

H_0: A amostra provém da distribuição escolhida;

H_1: A amostra não provém da distribuição escolhida.

Uma vantagem importante deste teste é que ele pode ser aplicado a qualquer distribuição uni-variada para a qual se possa calcular sua função de distribuição cumulativa. Este teste é aplicado para dados agrupados em classes. Neste ponto, cabe uma observação importante: o resultado do teste depende de como os dados foram agrupados. Outra desvantagem importante do teste qui-quadrado é a necessidade de uma amostra grande para que os resultados sejam válidos.

Para o cálculo do teste qui-quadrado, a seguinte estatística é calculada:

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} \tag{2.13}$$

O_i é a frequência observada para a classe *i* e E_i é a frequência esperada para a classe *i*, sendo calculada pela expressão:

$$E_i = n * (F(y_{ii}) - F(y_i)) \tag{2.14}$$

Onde *n* é o tamanho da amostra, *F* é a função cumulativa da distribuição que está sendo testada, y_{ii} é o limite superior para a classe *i* e y_i é o seu limite inferior. A definição da quantidade de classes é arbitrária e não há uma escolha ótima, já que esta escolha depende da distribuição.

A estatística χ^2 segue uma distribuição qui-quadrado com *k-p-1* graus de liberdade, onde *p* é o número de parâmetros que estão sendo estimados da amostra. A hipótese de que os dados provêm de uma população com os parâmetros especificados da distribuição é aceita desde que o valor de χ^2 seja menor do que o valor crítico obtido na tabela em função da significância α e *k-p-1*graus de liberdade.

2.3.3.2 Teste de Ajuste Kolmogorov-Smirnov

O teste de ajuste de Kolmogorov-Smirnov foi inicialmente proposto por Kolmogorov (1933) e Smirnov (1936). Diferentemente do teste qui-quadrado, que pode ser usado para testar distribuições contínuas e discretas, o teste K-S é apropriado para testar somente distribuições contínuas, como por exemplo, a distribuição *Weibull*. De acordo com Trivedi (2002), quando aplicado a distribuições discretas, o teste K-S produz resultados conservadores. Então, a probabilidade de se cometer um erro do tipo I (rejeitar a hipótese H_0 sendo esta verdadeira) será quando muito igual a α (nível de significância do teste). Entretanto, esta vantagem é compensada por um aumento na probabilidade de se cometer um erro do tipo II (aceitar H_0 sendo esta falsa).

O teste K-S consiste em ordenar a amostra dos dados de forma que $x_1 \leq x_2 \leq ... \leq x_n$. Então, os valores empíricos da função de distribuição amostral são dados por:

$$\hat{F}(x) = \begin{cases} 0 & x \leq x_1 \\ \dfrac{i}{n} & x_i \leq x \leq x_{i+1} \\ 1 & x_n \leq x \end{cases} \qquad (2.15)$$

Uma medida do desvio da função de distribuição cumulativa empírica (FDCE ou $\hat{F}(x)$) da função de distribuição cumulativa teórica (FDC ou F(x)) é definida por:

$$d_n(x) = \left| \hat{F}(x) - F(x) \right| \tag{2.16}$$

Como F(x) é conhecida, o valor do desvio pode ser calculado para cada valor de *x*. Entre todos os valores de d_n, o maior deles é uma indicativa de quanto os valores empíricos se distanciam dos valores teóricos. Como $\hat{F}(x)$ é uma função por partes (com *n* passos) e F(x) é uma função contínua e crescente, é suficiente calcular os valores de $d_n(x)$ nos pontos à esquerda e à direita de cada intervalo $[\, x_i, x_{1+1}\,]$. O valor máximo de $d_n(x)$ é, então, o valor da estatística Kolmogorov-Smirnov. O Quadro 2.3 mostra o procedimento para o cálculo do teste K-S para uma amostra de uma distribuição *Weibull*.

Observa-se que o valor máximo para os valores de $d_n(x)$ calculados é igual a 0.207. Usando a tabela dos valores da região crítica da estatística D_n, verifica-se que com um nível de significância de 5% (α=0.05), $D_{10,\alpha}$ = 0.41 e, portanto, a região crítica é $\{D_{10;\alpha} \geq 0.41\}$. Então, aceita-se a hipótese nula ao nível de 5%.

i	X_i	ECDF	CDF	D_+	D_-
1	0.1272	0.1	0.0160	0.0840	0.0160
2	0.3049	0.2	0.0888	0.1112	0.0112
3	0.4288	0.3	0.1680	0.1320	0.0320
4	0.5898	0.4	0.2938	0.1062	0.0062
5	0.6400	0.5	0.3361	0.1639	0.0639
6	0.7062	0.6	0.3927	**0.2073**	0.1073
7	1.1279	0.7	0.7198	0.0198	0.1198
8	1.2262	0.8	0.7776	0.0224	0.0776
9	1.4022	0.9	0.8600	0.0400	0.0600
10	2.3890	1	0.9967	0.0033	0.0967

Quadro 2.3 - Cálculo do teste K-S.

O teste Anderson-Darling (A-D) (LAW; KELTON, 2000, p. 368) é outra possibilidade de teste, diferenciando-se do K-S em virtude daquele ter sido projetado a fim de detectar discrepâncias nas caudas das distribuições.

2.3.3.3 Dificuldades no Ajuste de Distribuições

Diversos autores discutem as dificuldades e limitações dos testes de ajuste, como por exemplo, (VINCENT, 1998, p. 87), (LAW; KELTON, 2000, p. 356) e (KUHL et al., 2008, p. 49). Quando o tamanho da amostra é pequeno, os testes possuem baixa capacidade de detecção da falta de aderência entre os dados e as distribuições teóricas (poder do teste), acabando por não rejeitar qualquer alternativa de distribuição. Por outro lado, quando o tamanho da amostra é muito grande, uma discrepância insignificante dos dados em relação às distribuições teóricas faz com que os testes rejeitem todas as alternativas de distribuições.

Considerando estas dificuldades, alguns softwares fornecem o nível descritivo dos testes (*valor-p* = $P(Z \geq z \mid H_0)$) que é a probabilidade de, partindo-se do pressuposto que a hipótese H_0 é verdadeira, obter-se um resultado tão ou mais extremo que o verificado na amostra. Neste caso, sendo a hipótese H_0 verdadeira, então, a amostra em questão advém da distribuição teórica. Desta forma, quanto menor o *valor-p*, menos provável será que a amostra advenha da distribuição teórica. A hipótese H_0 será rejeitada ao nível de significância (α = 5%) se o *valor-p* calculado da amostra for menor que 0,05.

No item 2.3.3 procurou-se dar uma visão de como se faz o ajuste de uma amostra de uma variável aleatória a uma distribuição de probabilidade, bem como a determinação dos seus parâmetros. Para o leitor interessado em um maior aprofundamento deste assunto, recomenda-se (KENDALL; STUART, 1979), (SNEDECOR; COCHRAN, 1989), (KENDALL; STUART; ORD, 1994) e (LAW; KELTON, 2000).

2.4 GERAÇÃO DE SEQUÊNCIAS ALEATÓRIAS EM AMBIENTES DE SIMULAÇÃO

Conforme já foi mencionado, estabelecer qual é a melhor distribuição que se ajusta a uma amostra de dados é de fundamental importância para o analista de um sistema, independentemente da utilização de teoria de filas ou de simulação. Por outro lado, compreender as técnicas de geração de sequências aleatórias é necessário somente quando se opta pela simulação, sobretudo, se o projeto for implementado em um ambiente computacional que não disponha de funções de geração de sequências aleatórias.

Subjacente a qualquer ambiente de simulação estocástica, existe um gerador de números aleatórios (GNA). Sua finalidade é produzir uma sequência de números que representa as observações ou amostras de um fenômeno que, em última instância, simula a aleatoriedade existente no sistema real. Esta sequência deve ser contínua e estar uniformemente distribuída no intervalo [0,1] e, além disso, os valores devem ser independentes uns dos outros.

2.4.1 Geração de Números Aleatórios Independentes e Uniformemente Distribuídos

Um GNA implementado computacionalmente é, geralmente, um algoritmo recursivo que produz uma sequência de números a partir de uma semente (como todo algoritmo recursivo). Assim sendo, esta sequência não é verdadeiramente aleatória e, então, o GNA produz "pseudo" números aleatórios. Apesar deste fato, os algoritmos modernos são capazes de produzir uma sequência de números que são, na prática, aleatórios, passando com sucesso por testes estatísticos que demonstram que eles são independentes e estão uniformemente distribuídos entre [0,1].

2.4.1.1 Gerador congruencial linear (GCL)

Os GNAs mais populares e amplamente usados em linguagens de programação são, de alguma forma, casos especiais do método introduzido por Lehmer (1951), conhecido por Gerador Congruencial Linear (GCL). A sequência de números aleatórios é obtida pela seguinte expressão:

$$X_{n+1} = (a * X_n + c) \bmod m; \qquad n \geq 0$$
$$U_n = \frac{X_n}{m} \tag{2.17}$$

onde,

m, módulo	$0 < m$
a, multiplicador	$0 \leq a < m$
c, incremento	$0 \leq c < m$
$\mathbf{X_0}$ semente	$0 \leq X_0 < m$

A operação "*mod m*" significa dividir por *m* e retornar o resto da divisão. Os valores de *m, a* e *c* devem ser criteriosamente escolhidos, a fim de que a sequência seja realmente IID U[0,1]. Law & Kelton (2000, p. 418) apresentam os testes empíricos e teóricos que podem ser usados a fim de determinar se a sequência obtida por meio dos valores *a*, *m*, *c* e X_o são IID U[0,1].

Knuth (1997) observa que a existência de ciclos é comum a todas as sequências que tenham a forma geral $X_{n+1} = f(X_n)$, quando f transforma um conjunto finito em si mesmo. É desejável, portanto, que um bom GNA produza uma sequência aleatória com um ciclo ou período longo. A título de exemplo, no início de 1980, a implementação do GNA do *Arena Simulation* usava $m = 2^{31}-1$, $a = 7^5$ e $c = 0$. Neste caso, o comprimento do ciclo era 2.1 bilhões e, para aquela época, este tamanho de ciclo era aceitável. À medida que os computadores ficaram mais rápidos, esse tamanho de ciclo passou a não ser mais adequado, pois se uma simulação rodasse durante horas era provável que o ciclo se repetisse, produzindo resultados incorretos e, em consequência, prejudicando a validade do modelo.

Há um grande número de trabalhos desenvolvidos propondo GNAs que superam as limitações do GCL. A seguir será apresentado o GNA atualmente usado pelo *Arena Simulation.*

2.4.1.2 Gerador Recursivo Multiplicativo Combinado (GRMC)

O Gerador Recursivo Multiplicativo Combinado (GRMC) foi proposto por L´Ecuyer *et al.* (2002). Inicialmente, o gerador calcula duas recursões separadas:

$$\begin{aligned} A_n &= (1403580 * A_{n-2} - 810728 * A_{n-3}) \bmod 4294967087 \\ B_n &= (527612 * B_{n-1} - 1370589 * B_{n-3}) \bmod 4294944443 \end{aligned} \qquad (2.18)$$

Em seguida, estes dois valores são combinados de acordo com a expressão seguinte:

$$\begin{aligned} Z_n &= (A_n - B_n) \bmod 4294967087 \\ U_n &= \frac{Z_n}{4294967088}, \; se \; Z_n > 0 \; ou \\ U_n &= \frac{4294967087}{4294967088}, \; se \; Z_n = 0 \end{aligned} \qquad (2.19)$$

Para o passo inicial do gerador, é necessário um vetor com seis sementes, três para a geração dos primeiros três A_n's e três para os três primeiros B_n's. As constantes envolvidas no gerador foram cuidadosamente escolhidas sendo que L´Ecuyer (1999) apresenta as justificativas que garantem que as sequências aleatórias geradas são independentes e uniformemente distribuídas e, além disso, que o ciclo é da ordem de $3.1*10^{57}$. É óbvio que a velocidade do gerador GRMC é um pouco mais lenta do que a do GCL, o que não significa que isso seja um problema, visto que o poder computacional atual das máquinas é mais elevado que anteriormente.

Por fim, outro exemplo de GNA é o *Mersenne twister* (MATSUMOTO; NISHIMURA, 1998). O nome deste gerador provém do fato de que o comprimento do ciclo é um número primo *mersenne* (um número *mersenne* é um número 2^n-1, uma potência de 2 menos 1). Existem duas variantes deste método: *Mersenne Twister* MT19937 (com palavra de 32 bits) e MT19937-64 (com palavra de 64 bits). A variante *Mersenne Twister* MT19937 possui comprimento de ciclo de estrondosos 2^{19937}-1.

2.4.2 Geração de Números Aleatórios para Distribuições Teóricas

No item anterior, foi mostrado como se obtêm, computacionalmente, amostras de uma distribuição U~[0,1]. Entretanto, em ambientes de simulação, o projetista necessita de amostras das demais distribuições teóricas, tanto para as discretas quanto para as contínuas.

Considerando um exemplo, para uma distribuição discreta que tenha sua função de probabilidade (*fp*) definida como,

$$p(x) = P(X = x) = \begin{Bmatrix} 0.2 & para\ x = 2 \\ 0.5 & para\ x = 3 \\ 0.3 & para\ x = 4 \end{Bmatrix}, \tag{2.20}$$

a ideia consiste em dividir o intervalo [0,1] em subintervalos com os mesmos comprimentos especificados na *fp* ([0, 0.2), [0.2, 0.7), [0.7, 1]). Para cada número aleatório U gerado, verifica-se a qual subintervalo o número gerado pertence, atribuindo à variável X o valor correspondente.

A seguir serão apresentados os métodos para geração de sequências aleatórias para distribuições contínuas: transformação inversa, aceitação/rejeição, convolução e composição.

2.4.2.1 Método da Transformação Inversa

Supondo que se deseja obter uma amostra de uma variável *X~F* de uma determinada distribuição, o método da transformação inversa consiste em:

1) gerar um número randômico '*u*' ;

2) atribuir $x = F^{-1}(u)$. (2.21)

O seguinte teorema pode ser usado para provar que este método está correto (TRIVEDI, 2002):

Teorema 2.1: seja *X* uma variável aleatória com *fdp* f_x, que é diferente de zero em um subconjunto *I* ($f_x(x) > 0$, $x \in I$ e $f_x(x) = 0$, $x \notin I$). Seja Φ uma função monótona diferenciável, cujo domínio está em *I*. Então, $Y = \Phi(x)$ é uma variável aleatória com *fdp* f_y, *igual a:*

$$f_y(y) = \begin{cases} f_x\left[\Phi^{-1}(y)\right]\left|\Phi^{-1'}(y)\right| & , \quad y \in \Phi(I) \\ 0, & y \notin \Phi(I) \end{cases} \tag{2.22}$$

Φ^{-1} é a função inversa univocamente determinada de Φ e $(\Phi^{-1})'$ é a derivada da função inversa.

Prova do Teorema 2.1: o teorema pode ser provado assumindo que Φ(x) é uma função crescente de *x*. Sendo $Y = \Phi(x)$, então:

$$\begin{aligned} F_y(y) &= P(Y \le y) = P(\Phi(X) \le y) \\ F_y(y) &= P(X \le \Phi^{-1}(y)), \text{ visto que } \Phi(x) \text{ é monótona crescente} \\ F_y(y) &= F_x(\Phi^{-1}(y)) \end{aligned} \tag{2.23}$$

Derivando ambos os lados e usando a regra da cadeia, obtém-se (2.22). Para provar que o método apresentado em (2.21) está correto será usado o teorema 2.1.

Prova do Método (2.21)**:** seja Φ uma FDC *F* de uma variável aleatória *X*, com *fdp* f_x. Aplicando o teorema 2.1:

$$\begin{aligned} &\mathrm{Y} = \mathrm{F}(\mathrm{X})\text{, então} \\ &\mathrm{F_y}(\mathrm{y}) = F_x(F_x^{-1}(y)) \\ &\mathrm{F_y}(\mathrm{y}) = y \end{aligned} \tag{2.24}$$

Portanto, a variável aleatória Y tem *fdp* igual a:

$$\mathrm{f_y}(\mathrm{y}) = \begin{Bmatrix} 1, 0 < \mathrm{y} < 1 \\ 0, \text{senão} \end{Bmatrix} \tag{2.25}$$

Desta forma, se *X* é uma variável aleatória contínua com função cumulativa *F*, então a nova variável aleatória *Y=F(X)* é uniformemente distribuída no intervalo (0,1). Esta ideia pode ser usada, então, para se produzir uma amostra de *X*, gerando primeiro um número aleatório *u* de uma distribuição uniforme e, em seguida, obtendo *x* por meio da relação $x=F^{-1}(u)$.

Supondo que se queira uma amostra de uma variável $X \sim E(\lambda)$, cuja FDC é igual a $F(x)=1 - e^{-\lambda x}$, então:

$$\begin{aligned} &\mathrm{u} = 1 - \mathrm{e}^{-\lambda \mathrm{x}} \\ &\mathrm{e}^{-\lambda \mathrm{x}} = 1 - \mathrm{u} \\ &\ln^{\mathrm{e}^{-\lambda \mathrm{x}}} = \ln(1 - \mathrm{u}) \\ &\mathrm{x} = -\ln(1 - \mathrm{u}) / \lambda \\ &\mathrm{x} = -\ln(\mathrm{u}) / \lambda \quad -> \quad \mathrm{X} \sim \mathrm{E}(\lambda) \end{aligned} \tag{2.26}$$

O método da transformação inversa pode ser usado para se gerar facilmente sequências aleatórias de distribuições cuja FDC possua forma fechada[4] (*closed form*) de sua inversa, como por exemplo, exponencial, weibull, pareto, triangular e exponencial. O Quadro 2.4 apresenta as funções inversas para algumas distribuições:

Nome	fx,(x)	F(X)	**Função Inversa**
Exponencial	$\lambda e^{-\lambda x}, x > 0$	$1-e^{-\lambda x}, x > 0$	$x = -\frac{\ln(u)}{\lambda}$
Weibull	$\lambda \alpha x^{\alpha-1} e^{-\lambda x^{\alpha}}, x > 0$	$1-e^{-\lambda x^{\alpha}}, x > 0$	$x = \left(-\frac{\ln(u)}{\lambda}\right)^{1/\alpha}$
Pareto	$\alpha k^{\alpha} x^{-\alpha-1}, x > k$ $\alpha, k > 0$	$1 - \left(\frac{k}{x}\right)^{\alpha}$	$x = \frac{k}{u^{\frac{1}{\alpha}}}$
Cauchy	$\frac{\sigma}{\pi(x^2+\sigma^2)}$	$\frac{1}{2} + \frac{1}{\pi}\arctan(\frac{x}{a})$	$\sigma \tan(\pi u)$

Quadro 2.4 - Função inversa de distribuições teóricas

Há outras distribuições, como por exemplo, *normal*, *t de Student* e *Qui-Quadrado* para as quais não há uma forma fechada para sua FDC, mas há boas aproximações numéricas disponíveis, como mostram os trabalhos de Bratley, Fox & Schrage (1987), Gentle (2003) e Marsaglia, Zaman & Marsaglia (1994).

2.4.2.2 Método da Aceitação/Rejeição

Outra possibilidade para as distribuições cujas inversas de suas FDCs não possuam uma forma analítica é usar o método da *aceitação/rejeição*.

Supondo uma *fdp* $f_{x,}$ definida no domínio [a,b] e sabendo-se que seu valor máximo é igual a *k*, então, $0 \leq f_{x,}(x) \leq k$. O método funciona da seguinte maneira:

[4] Uma expressão é dita estar na forma fechada se é expressa analiticamente em termos de um número limitado de funções matemáticas. Assim, séries de potências não são permitidas.

1) Gerar um número aleatório uniforme *x* no intervalo [a,b];

2) Gerar um número aleatório uniforme *y* no intervalo [0,k];

3) Se y ≤ $f_{x,}$(x), então, retornar o valor de *x*. Senão, voltar ao passo 1.

Assim sendo, o método da aceitação/rejeição é um método de tentativa e erro. Grosso modo, gera-se um ponto no plano (x,y) no qual a *fdp* está plotada. Se o ponto estiver abaixo do gráfico da *fdp* ele será, então um ponto da distribuição $f_{x.}$ Caso contrário, deve-se tentar novamente.

O desempenho deste método é bastante sensível ao tamanho da área acima da $f_{x.}$. Portanto, na prática, usa-se uma função *g(x)* tal que $f_{x.}(x) \leq c\ g(x)$, procurando-se minimizar a probabilidade de erro no passo 3. A função *g(x)* é conhecida por função de majorização (*hat function* ou *majoring function*) (GENTLE; HÄRDLE; MORI, 2004).

2.4.2.3 Método da Convolução

Seja uma variável aleatória $X = Y_1 + Y_2 + ... + Y_n$, ou seja, *X* é igual a soma de *n* variáveis aleatórias independentes. O método da convolução consiste em gerar *n* variáveis independentes Y_i (por meio dos métodos anteriores) e somá-los para gerar *X*. Exemplos de variáveis aleatórias que podem ser expressas pela soma de outras são a hipoexponencial (soma de duas variáveis exponenciais com $\lambda_1 \neq \lambda_2$) e a k-Erlang (soma de *k* variáveis exponenciais com mesmo parâmetro λ)

2.4.2.4 Método da Composição

Este método se aplica quando a função de distribuição é uma combinação de outras funções de distribuição, como por exemplo:

$$\mathrm{F(x)} = \sum_{\mathrm{j=1}}^{\mathrm{m}} \mathrm{p_j} F_{\mathrm{j}}(x),\ p_j > 0\ e\ \sum p_j = 1, \tag{2.27}$$

Neste caso, pode-se aplicar o seguinte algoritmo:

1) Gerar um número inteiro aleatório, tal que $P(J=j)=p_{j,}$, para j=1...m;

2) Retornar X com a distribuição F_j.

2.5 GERAÇÃO DE ENTIDADES PARA PROCESSOS NÃO ESTACIONÁRIOS

No item anterior, foram apresentados os métodos numéricos frequentemente utilizados para a geração de sequências aleatórias para processos estocásticos estacionários. Entretanto, quando se modelam sistemas reais, ignorar a não-estacionariedade inerente a esses sistemas pode conduzir a resultados incorretos. Neste item, serão apresentados os métodos numéricos para a geração de sequências aleatórias para processos não estacionários.

2.5.1 Processos Poisson Não estacionários (Não-Homogêneos)

Basicamente, encontram-se na literatura três métodos para a geração de sequências aleatórias para processos Poisson não-homogêneos: **thinning** (LEWIS; SHEDLER, 1979 apud HARROD; KELTON, 2006), **inversão da função de taxas cumulativa** (*inversion of cumulative rate function*) (ÇINLAR, 1975) e **composição** (*composition*) (DEVROYE, 1986). Os métodos *thinning* (ver Figura 2.8) e *composition* requerem a geração de dois números aleatórios U~[0,1] para cada evento, exigindo portanto maior tempo computacional. Recomenda-se (DEVROYE, 1986) para leitores interessados no aprofundamento desses dois métodos.

O método de *thinning* parte do pressuposto da existência de uma função conhecida e inversível $\mu(t)$, de forma que $\lambda.(t) \leq \mu(t)$, para todo t. $\lambda.(t)$ é a função que define a taxa do processo *Poisson* não-homogêneo.

Thinning Algorithm

```
1  T = 0
2  k = 0
3  Repita
4    Gere Z, primeiro evento em processo
     Poisson com taxa μ ocorrendo depois de T.
     Atribua T = Z
5    Gere um número U~[0,1]
6    Se U ≤ λ(Z)/μ(Z), então
7          k = k + 1
8          X_k = T
9    Fim Se
10 Até parar
```

Figura 2.8 - Algoritmo *Thinning*

Fonte: (DEVROYE, 1986).

Os valores de X_k´s representam um processo Poisson não-homogêneo com taxa definida pela função λ.. O que o algoritmo faz é produzir uma sequência $0<Y_1<Y_2<...$ com taxa definida pela função μ e eliminando alguns pontos, num processo muito semelhante ao método da aceitação/rejeição.

Quando possível, recomenda-se o método da inversão devido à sua simplicidade. (HARROD; KELTON, 2006) apresenta três algoritmos, baseados no método da inversão, para a geração de sequências aleatórias para processos Poisson não-homogêneos (observe que os autores usam o termo não-estacionário como sinônimo de não-homogêneo): *Direct Algorithm*, *First Walkup Algorithm* e *Second Walkup Algorithm* (ver Quadro 2.5).

Para os três algoritmos, é definida uma função cumulativa de chegada $\Lambda(t)$ para o processo Poisson não-homogêneo, conforme mostra a Figura 2.9, e um processo estacionário Poisson com taxa λ=1. Cada evento do processo estacionário (S_1, S_2, **...**) possui intervalos entre eventos $A_i = S_i - S_{i-1}$, sendo $S_0=0$. Os eventos do processo Poisson não-homogêneo (T_1, T_2, **...**) são obtidos pela expressão $T_n = \Lambda^{-1}\left(\sum_{i=1}^{n} A_i\right) = \Lambda^{-1}(S_n)$. Os eventos do processo estacionário são calculados gerando os valores $A_i=-ln(u)$ (ver item 2.4.2.1) com taxa igual a 1.

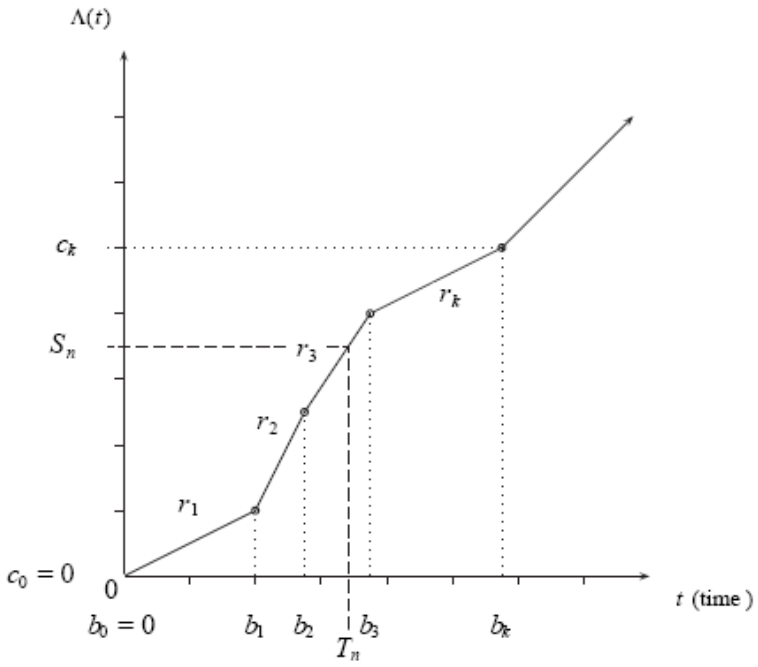

Figura 2.9 - Função cumulativa de chegada - Λ(t)

Fonte: (HARROD; KELTON, 2006)

A função cumulativa $\Lambda(t)$ é contínua e linear por partes, com taxa de chegada em cada segmento *k* igual a r_k (λ(t) é igual a r_k em cada segmento *k*). Cada evento S_n é representado no eixo vertical e os eventos T_n são mapeados, pela função inversa Λ^{-1}, no eixo horizontal.

Direct Algorithm	*First Walkup Algorithm*
1 n = 1 2 T_n = 0 3 k = 1 4 S_{n-1}= 0 5 **Repita** 6 Gere u=U~[0,1] 7 A_n = −ln(u) 8 S_n = S_{n-1} + A_n 9 **Enquanto** S_n − c_k > 0 **faça** 10 k = k + 1 11 **Fim enquanto** 12 T_n = b_{k-1}+(S_n−c_{k-1})/r_k 13 Exibir T_n 14 n = n + 1 15 **Até** parar	1 n = 1 2 T_{n-1} = 0 3 k = 1 4 S_{n-1} = 0 5 **Repita** 6 Gere u=U~[0,1] 7 A_n = −ln(u) 8 **Enquanto** S_{n-1} + A_n − c_k > 0 **faça** 9 T_{n-1} = b_k 10 A_n = A_n − (c_k − S_{n-1}) 11 S_{n-1} = c_k 12 k = k + 1 13 **Fim enquanto** 14 W_n = A_n/r_k 15 T_n = T_{n-1} + W_n 16 S_n = S_{n-1} + A_n 17 Exibir T_n 18 n = n + 1 19 **Até** parar

Second Walkup Algorithm
1 n = 1 2 T_{n-1} = 0 3 k = 1 4 **Repita** 5 Gere u=U~[0,1] 6 A_n = −ln(u) 7 W_n = A_n/r_k 8 **Enquanto** T_{n-1} + W_n − b_k > 0 **faça** 9 A_n = A_n − (r_k(b_k − T_{n-1})) 10 T_{n-1} = b_k 11 k = k + 1 12 W_n = A_n/r_k 13 **Fim enquanto** 14 T_n = T_{n-1} + W_n 15 Exibir T_n 16 n = n + 1 17 **Até** parar

Quadro 2.5 – Algoritmos *Direct*, *First Walkup* e *Second Walkup*

Fonte: (HARROD; KELTON, 2006)

Harrod e Kelton (2006) não mencionam, entretanto, que o mapeamento só funciona corretamente desde que a diferença entre os segmentos seja igual à unidade ($b_k - b_{k-1} = 1$). Caso o tamanho do segmento seja diferente da unidade, nos três algoritmos é necessário proceder à seguinte correção:

Direct Algorithm
12 $T_n = b_{k-1} + (b_k - b_{k-1}) * (S_n - c_{k-1}) / r_k$
First Walkup Algorithm
15 $T_n = T_{n-1} + W_n * (b_k - b_{k-1})$
Second Walkup Algorithm
14 $T_n = T_{n-1} + W_n * (b_k - b_{k-1})$

Para os três algoritmos, o procedimento de mapeamento do processo estacionário (S_n) para o processo de Poisson não homogêneo (T_n), dentro do mesmo segmento, é feito de forma similar. A diferença entre eles é o momento de se fazer a transição de um segmento para outro. O algoritmo direto apenas muda de segmento quando o evento estacionário ultrapassa o valor de c_k. Os outros dois algoritmos são chamados de "*walkup*" porque, em cada transição, eles passam pelo ponto de quebra do segmento b_k. Apesar do procedimento adotado na transposição entre segmentos ser o mesmo tanto para o *First Walkup* quanto para o *Second Walkup*, este não necessita da variável S_n. No entanto, os autores não mencionam, no seu trabalho, um aspecto negativo (verificado quando da implementação destes métodos durante esta pesquisa) dos métodos *First Walkup* e *Second Walkup*. O último evento gerado em cada segmento cairá sempre no instante da intersecção dos segmentos. Quando a taxa de chegada r_k for muito pequena (< 1), como, em média, tem-se menos que um evento por segmento, este evento vai ocorrer sempre nas intersecções, o que pode produzir resultados insatisfatórios na modelagem de sistemas reais.

2.5.2 Processos de Renovação

Assim como é necessário tratar a não estacionariedade nos processos Poisson, diversos estudos nas áreas de telecomunicações, redes de computadores e sistemas de manufatura mostram que os processos de chegada podem não ser rigorosamente Poisson. Paxson & Floyd (1995) analisaram 24 *traces* (captura de pacotes em monitoramento de redes) em uma rede de longa distância e mostram que, para determinados pacotes de rede, considerar os intervalos de chegada como exponenciais não modela corretamente o efeito de rajada (*burst*) existente nestes protocolos. Kelton (2007, p.39) adverte que, apesar de alguns

softwares de simulação poderem representar processos NSPP, o mesmo não acontece quando estruturas estocásticas não estacionárias mais genéricas fazem parte do problema.

Gerhardt & Barry (2009, p. 2) apresentam um algoritmo (exibido no Quadro 2.6) para a geração de um PR generalizado, não-estacionário e não-Poisson usando o método da inversão. Para o processo não-estacionário, *r(t)* denota a *função taxa* e $R(t)=\int_0^t r(u)du$ a *função taxa integrada*, assumindo-se que *r(t)* seja integrável. As funções de distribuição G_e e G são as distribuições da variável aleatória X_1 e das variáveis aleatórias X_i para i = 2,3, **...**, respectivamente, sendo $G_e \neq G$. Se G_e for igual a G, tem-se, então, um PR ordinário (GERHARDT ; BARRY, 2009, p. 2).

```
1   V0 = 0
2   n = 1
3   Gerar S1~Ge
4   V1 = R^-1(S1)
5   Repita
6       Wn = Vn - Vn-1
7       n = n + 1
8       Gerar Xn~G
9       Sn = Sn-1 + Xn
10      Vn = R^-1(Sn)
11  Até parar
```

Quadro 2.6 – Algoritmo para geração para um PR não-estacionário pelo método da inversão

Fonte: Baseado em (GERHARDT ; BARRY, 2009)

As distribuições G_e e G devem ser especificadas de forma que tenham taxa 1 e, desta forma, a sequência $\{W_n, n \geq 1\}$ representa os intervalos entre eventos da realização do processo estocástico não-estacionário.

Gerhardt & Barry (2009, p. 3) apresentam, também, um algoritmo que gera os tempos de intervalos entre chegadas pelo método de *thinning*, conforme mostra o Quadro 2.7 . O algoritmo inicia determinando o valor de $r^* \geq \max_{t\geq 0} r(t)$, que se assume sendo um valor finito. Desta forma, a sequência Y_k produzida é uma realização dos intervalos de chegada para o processo não estacionário gerado pelo método de *thinning*. Sendo que $\{M(t), t \geq 0\}$ denota o processo de contagem gerado pelo método, os autores provam que $E[M(t)] = R(t)$.

1	$k = 1$
2	$n = 1$
3	$T_0 = 0$
4	Gerar $S_1 \sim G_e$
5	Gerar $U_1 \sim U[0,1]$
6	**Se** $U_1 \leq r(S_1)/ r^*$ **então**
7	$T_1 = S_1$
8	$Y_1 = T_1 - T_0$
9	$k = 2$
10	**Fim se**
11	$n = n + 1$
12	Gerar $X_n \sim G$
13	$Sn = S_{n-1} + X_n$.
14	Gerar $U_n \sim U[0,1]$
15	**Se** $U_n \leq r(S_n)/ r^*$ **então**
16	$T_k = S_n$
17	$Y_k = T_k - T_{k-1}$
18	$k = k + 1$
19	**Fim se**
20	**Vá para** 11

Quadro 2.7 – Algoritmo para geração para um PR não-estacionário pelo método de *thinning*

Fonte: Baseado em (GERHARDT ; BARRY, 2009)

3 MÉTODOS PARA GERAÇÃO DE ENTIDADES, EM AMBIENTES DE SIMULAÇÃO, PARA PROCESSOS DE RENOVAÇÃO ORDINÁRIOS E NÃO ESTACIONÁRIOS

Neste capítulo, serão apresentados os métodos propostos para geração de entidades, em ambientes de simulação, para PRs ordinários e não estacionários. Com a finalidade de exemplificar como estes métodos podem ser implementados computacionalmente, fez-se uso do *Arena Simulation* o qual sabidamente possui ferramentas e funções previamente desenvolvidas para facilitar o desenvolvimento de projetos de simulação. No entanto, caso os métodos propostos sejam utilizados em ambientes computacionais nos quais não haja ferramentas adequadas para simulação, faz-se necessário, antecipadamente ao desenvolvimento dos métodos, a criação destas funções, como por exemplo, a geração de números aleatórios – amplamente discutidas na revisão bibliográfica.

Considerando a diversidade de situações para aplicação de PRs não estacionários existentes no mundo real, esta tese propõe subdividi-los em três classes de problemas:

1) PRs com taxa constante por período, de uma origem para múltiplos destinos;
   São processos cuja taxa de chegada é constante por período e são originados de um único ponto ou fonte e destinam-se a múltiplos destinos. Exemplo deste tipo de processo é uma central de atendimento ao público, na qual as chamadas são originadas, exclusivamente por linhas telefônicas.

2) PRs com taxa constante por período, de múltiplas origens para múltiplos destinos;
   São processos cuja taxa de chegada é constante por período e são originadas de várias fontes ou origens para múltiplos destinos. Exemplo: em um modelo de simulação de uma rede de computadores, cada pacote ou mensagem, que trafega na rede, possui uma origem e um destino próprios. Como uma rede de computadores é por definição composta de vários nodos, então há múltiplas origens e múltiplos destinos.

3) PRs com taxas definidas por meio de planejamento;
São processos cujas taxas de chegada são estabelecidas por meio de uma demanda futura a ser atingida. Exemplo: em um sistema logístico, a quantidade diária que é transportada de uma determinada origem para um destino é definida em função da quantidade total necessária no destino em uma data futura e da capacidade de recebimento/tratamento destas mercadorias.

A fim de facilitar, ao leitor, a compreensão dos métodos expostos a seguir, sugere-se a leitura dos próximos itens com a leitura simultânea da aplicação dos métodos, no próximo capítulo.

3.1 PROCESSOS DE RENOVAÇÃO COM TAXA CONSTANTE POR PERÍODO, DE UMA ORIGEM PARA MÚLTIPLOS DESTINOS

São PRs cuja taxa de chegada é constante por período e as entidades são originadas de uma única fonte para múltiplos destinos. Como exemplo deste tipo de processo não-estacionário, pode-se considerar um modelo de simulação de uma central de atendimento ao público (*call center*), no qual as chamadas são originadas por meio de linhas telefônicas (única origem das entidades) e são classificadas ou destinadas internamente por tipo de chamada - vendas, situação das ordens de compra, suporte pré-venda, suporte pós-venda, assistência técnica, entre outros – (múltiplos destinos). Quando simulação é a técnica escolhida para modelar estes sistemas, em regra, primeiro geram-se as entidades de chamadas e, em seguida, elas são classificadas por um critério de percentual de acordo com o tipo de chamada. No método proposto, as entidades quando entram no modelo já chegam previamente classificadas.

3.1.1 Definindo as Variáveis do Problema

Seja um PR não-estacionário para o qual se tenha os seguintes dados de entrada:

- ***P***: quantidade de períodos ou faixas na simulação;
- ***C***: quantidade de classes definida para o processo na simulação;

- $\boldsymbol{NC_c}$: indica o nome da classe *c*;

- $\boldsymbol{R_{c,p}}$: é a taxa de chegada na simulação para a classe *c* no período *p*;

- $\boldsymbol{B_p}$: é o valor temporal na simulação para cada período *p*, sendo $B_0 = 0$;

- **Φ**: inversa da função cumulativa (função de distribuição) - e seus parâmetros - do processo a modelar. Esta função define o ajuste (distribuição) que será feito nos intervalos de chegadas para o conjunto de valores gerados pelo algoritmo. Caso seja um processo NSPP, **Φ** deve ser igual a *–ln(u)*, onde *'u'* (ver item 2.4.2.1) é um número aleatório uniformemente distribuído entre [0,1]. Neste caso, a sequência de valores produzidos por **Φ** estará exponencialmente distribuída com média e taxa igual a 1;

Com estes dados de entrada, deseja-se obter um vetor T_i, $i = 1...n$, sendo que T_i representa o i-éssimo evento a ser gerado na simulação. Ressalte-se, entretanto, que *'i'* não representa o tempo de chegada, mas tão somente a ordem dos eventos. Ou seja, o tempo de chegada de T_i é menor ou igual ao tempo de chegada de T_{i+1}. Para cada evento *'i'*, T_i indica o momento temporal da chegada e a qual classe refere-se o evento.

Para a solução deste problema, foi utilizada uma versão modificada do algoritmo *Direct Algorithm* (ver item 2.5.1) adaptando-o no que necessário para a geração de eventos para múltiplas classes. Desta forma, fez-se uso, ainda, das seguintes variáveis:

- $\boldsymbol{C_{c,p}}$: é a soma, para uma determinada classe *c*, das taxas de chegada de todos os períodos até p. $C_{c,p} = \sum_{i=1}^{p} R_{c,i}$

- $\boldsymbol{A_i}$: sendo $\Lambda(t)$ (ver Figura 2.9) a função cumulativa dos intervalos de chegada para um processo estacionário definido pela função **Φ,** com média igual a 1, então, $S_1, S_2, ... S_n$, são os tempos de chegada dos eventos e $A_i = \boldsymbol{\Phi}$ e $S_i = S_{i-1} + A_i$;

- $\boldsymbol{S_n}$: tempos dos eventos para o processo estacionário definido pela função **Φ**, sendo $S_o = 0$, então, $S_n = \sum_{i=1}^{n} A_i$;

➢ Obs: Durante a execução do algoritmo proposto, não é necessário manter todos os valores gerados tanto para vetor *A* quanto para *S***.** Desta forma, no algoritmo proposto, *A* e *S* não precisam ser vetores. S_n é o somatório dos últimos *n* intervalos de chegada dados por A_i.

- ***$T_{c,i}$*:** tempos de chegada para o processo não-estacionário da classe *c*. Enquanto que os eventos A_i (estacionários) estão representados no eixo *Y*, os eventos $T_{c,i}$ são representados no eixo *X* e são gerados pela transposição da equação da reta, dada por $R_{c,p}$.

3.1.2 Algoritmo de Geração dos Eventos

O Quadro 3.1 apresenta o algoritmo principal que gera o vetor *T* cujo conteúdo são todos os eventos do processo não-estacionário para todas as classes em consonância com as taxas definidas por período e por classe em $R_{c,p}$.

```
Início Algoritmo Principal
1       Obtenha os dados de entrada (P, C, NC_c, R_c,p, B_p e Φ)
2       Para c = 1 até C faça
3               Para p = 1 até P faça
4                       C_c,p= C_c,p-1 + R_c,p
5               Fim para
6       Fim Para
7       Para c = 1 até C faça
8               Chame DirectAlgorithmporClasse(T_c)
9       Fim para
10      Classifique, por ordem de chegada, todos os vetores T_c´s e mescle-os em um único
        vetor T no qual cada entrada T_i tenha o formato: tempo de chegada, c, NC_c.
11      Retorne T
Fim
```

Quadro 3.1 – Algoritmo *Principal* PR com taxas constantes por período, de uma origem para múltiplos destinos

Depois de obter, na linha 1, todos os dados de entrada do problema, o algoritmo principal totaliza em $C_{c,p}$, na linha 4, para cada classe e período, os valores das taxas de chegada $R_{c,p}$. Na linha 8, o algoritmo chama para cada classe o procedimento

DirectAlgorithmporClasse (Quadro 3.2), o qual constrói o vetor T_c cujo conteúdo é o conjunto de eventos gerados para a classe *c*. Na linha 10, todos os vetores T_c´s são mesclados em um único vetor T, que é o resultado esperado do algoritmo.

```
Início DirectAlgorithmporClasse (Tc)
1       Dimensione o vetor Tc com o tamanho Cc,P
2       n = 0
3       p = 1
4       Sn = 0
5       Faça
6               n = n + 1
7               Gere um número aleatório u~U[0,1]
8               A = Φ (u)
9               Sn = Sn + A
10              Se Sn < Cc,P faça
11                      Enquanto (Sn – Cc,p) > 0 faça
12                              p = p + 1
13                      Fim enquanto
14                      Se n > Cc,P faça
15                              Redimensione o vetor Tc,n para o tamanho n
16                      Fim se
17                      Tc,n = Bp-1 + ( (Sn – Cc,p-1) / Rc,p ) * (Bp – Bp-1)
18              Fim se
19      Fim faça enquanto Sn < Cc,P
20      Retorne Tc
Fim
```

Quadro 3.2 – Algoritmo *DirectAlgorithmporClasse* para PR com taxas constantes por período, de uma origem para múltiplos destinos

O procedimento *DirectAlgorithmUpporClasse* é baseado no algoritmo apresentado no item 2.5. No entanto, algumas modificações importantes foram feitas a fim de adaptá-lo aos objetivos definidos.

Uma primeira modificação importante, para fins de eficiência computacional, foi em relação ao tamanho do vetor T_c. Como as taxas de chegada podem ser muito diferentes para cada classe, o total de eventos gerados por classe também poderá ser bastante diferente. Assim sendo, o procedimento inicia definindo, na linha 1, o tamanho deste vetor para $C_{c,P}$ que é o valor esperado para o total de eventos na classe *c*. Caso sejam gerados mais eventos do que o esperado (teste efetuado na linha 14), este vetor é aumentado em uma posição. Então, ao término do procedimento, quando muito, no caso de o total dos eventos gerados estar abaixo do valor esperado, poucas posições do vetor estarão sem uso.

Outra modificação foi em relação à função definida para o processo estacionário no eixo *Y*. Neste caso, a inversa da função de distribuição não é fixa e sim um parâmetro do algoritmo, permitindo que se modelem outras formas de PRs não estacionários. Como última consideração importante, o algoritmo foi alterado para poder tratar múltiplas classes de eventos.

3.1.3 Implantação do algoritmo no *Arena Simulation*

Com a finalidade de testar e validar o algoritmo proposto, sua implementação foi feita no *Arena Simulation* usando o recurso do próprio ambiente chamado *template*. A construção de um *template* no *Arena* é feita, quase que na totalidade, por meio de três janelas: visão do usuário (*user view*), projeto da caixa de diálogo (*dialog design*) e lógica (*logic*).

A janela 'visão do usuário' é muito simples e apenas define como será o aspecto visual, para o usuário, do componente dentro do *template* e dispensa maiores comentários. A seguir são apresentadas as outras duas janelas.

3.1.3.1 Janela da caixa de diálogo

Quando o *template* é adicionado a um modelo de simulação, uma janela pode ser exibida ao projetista a fim de que ele possa fornecer alguns parâmetros necessários para a execução do componente. Para o método proposto, a caixa de diálogo, conforme mostra a Figura 3.1, é exibida para que o projetista possa indicar qual é o nome do arquivo que contém os dados de entrada do processo não-estacionário.

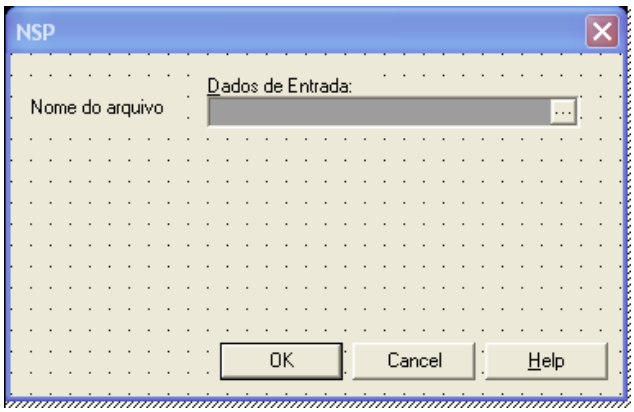

Figura 3.1- Caixa de diálogo do *template.*

3.1.3.2 Janela da lógica de programação

A Figura 3.2 mostra os componentes que fazem parte da lógica de programação do *template* criado.

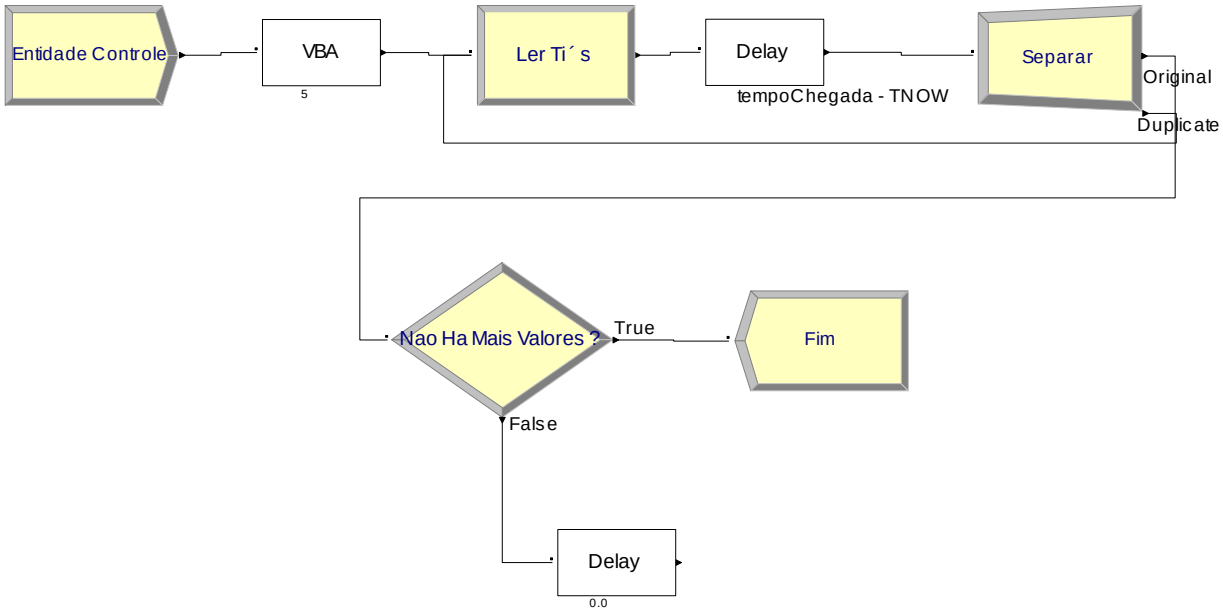

Figura 3.2 - Lógica de programação do template para o modelo não-estacionário (uma origem-múltiplos destinos)

O componente '*Entidade Controle*' cria uma única entidade de controle no início da execução do *template*. Esta entidade passa, em seguida, pelo componente '*VBA'* (implementado em *Visual Basic Application*) que executa o *Algoritmo Principal* e o *DirectAlgorithmporClasse*, descritos anteriormente. Ao término da execução destes procedimentos, o componente '*Ler Ti´s*', conforme mostra a Figura 3.3, lê todas as linhas de um arquivo texto cujo conteúdo é o tempo de chegada da próxima entidade e sua classe, armazenando estes valores nos atributos (*tempoChegada*, *iTipoClasse* e *nomeClasse*) da entidade.

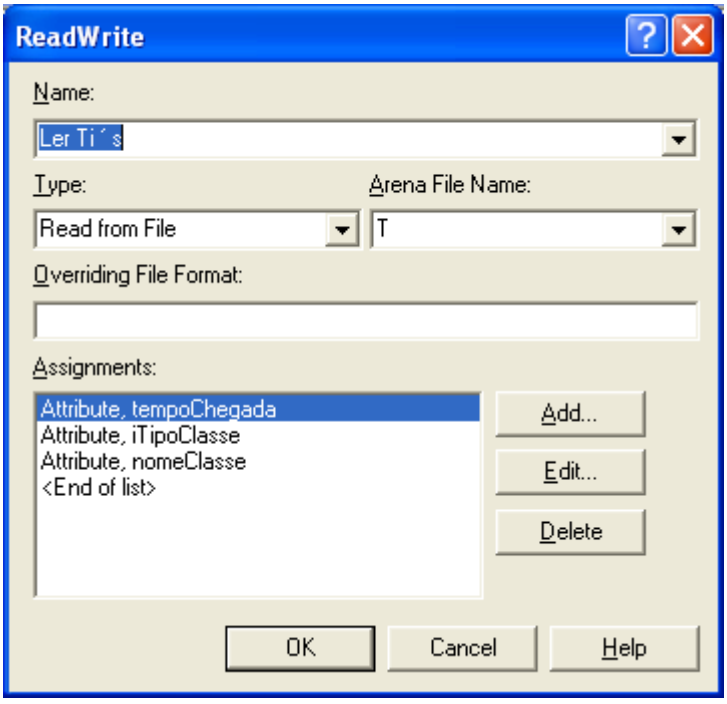

Figura 3.3 – Caixa de diálogo do componente *Ler Ti´s*

Após a leitura de uma linha do arquivo texto, o componente 'D*elay'* aguarda até o tempo de chegada indicado no atributo *tempoChegada* dado pela expressão (*tempoChegada* - *TNOW*). *TNOW* é uma variável interna do *Arena* que indica o tempo atual da simulação. No exato instante da chegada da entidade, o componente '*Separar'* divide a '*Entidade Controle*' em duas: uma volta para o componente '*Ler Ti´s´* e a outra vai para o componente *´Nao Ha Mais Valores*?'. Este verifica se a última linha do arquivo foi lida. Se isto ocorrer, o *template* é encerrado. Caso contrário, a entidade é encaminhada para o próximo componente 'Delay' que a encaminha para o ponto de saída (*exit point*) do template, conforme mostra a Figura 3.4. Observe que apesar de este ser um módulo de espera, a duração da espera é de zero, visto que este componente é usado apenas para encaminhar a entidade para o ponto de saída do *template* indicado no campo '*Next Label*'.

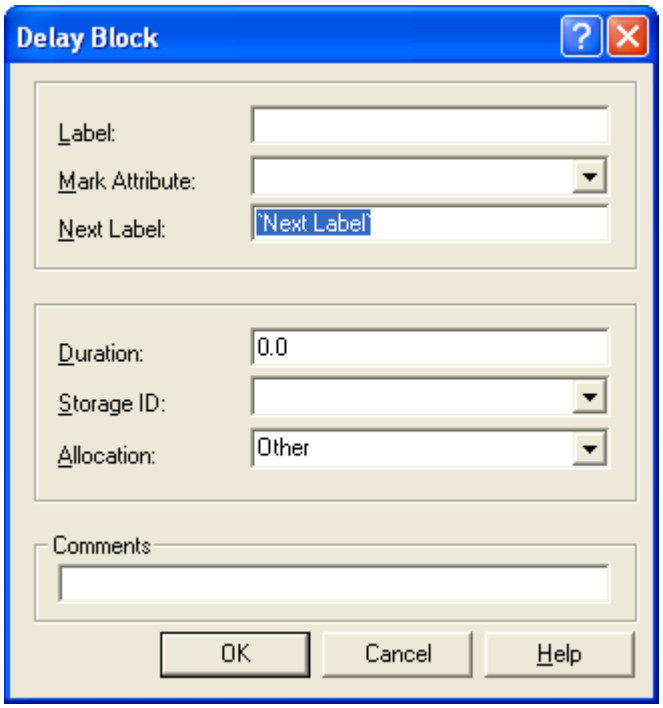

Figura 3.4 – Caixa de diálogo do componente que encaminha a entidade para o ponto de saída

3.1.4 Resultados do Algoritmo Proposto

Para tornar mais claro ao leitor, apresenta-se um exemplo numérico com os seguintes dados de entrada:

- $\boldsymbol{P}$ = 4;
- $\boldsymbol{C}$ = 3;
- $\boldsymbol{NC_c}$ = { ‘vendas’, ‘suporte’, ‘assistência técnica’}
- $R_{c,p}$ =

Classe	Períodos			
	1	2	3	4
1	200	500	100	410
2	500	800	10	200
3	100	300	100	200

- $\boldsymbol{B_p}$ = { 0, 30, 60, 90, 120};
- $\boldsymbol{\Phi}$ = $-ln(u)$;

Para este exemplo, tem-se então um processo NSPP. O período p = 1 vai do instante 0 até 30, o período p = 2 vai do instante de tempo maior que 30 até 60 e assim sucessivamente. De acordo com $R_{c,p}$, deverão ocorrer, em média, 10 eventos para a classe 2 no período *3*, com intervalos de chegadas distribuídos exponencialmente, dados pela função Φ**.**

Executou-se uma simulação[5] com 1000 rodadas[6] e os seguintes resultados foram obtidos, conforme ilustra o Quadro 3.3.

Totais por classe e período					
Classe	*Período*	*Média*	*HW*	*Mínimo*	*Máximo*
1	1	199,57	0,86418	157	249
1	2	500,48	1,3688	429	580
1	3	99,831	0,61124	70	143
1	4	409,49	1,2328	344	489
2	1	498,59	1,4058	430	569
2	2	798,59	1,7595	691	880
2	3	10,053	0,19655	2	24
2	4	200,2	0,92981	150	252
3	1	99,455	0,62421	71	137
3	2	299,52	1,0462	245	352
3	3	100,52	0,61556	67	130
3	4	199,94	0,83682	162	242
		Total de Eventos por rodada			
		3417,2	3,6807	3240	3637

Quadro 3.3 - Totais de eventos gerados por classe e por período com IC em 95% para 1000 rodadas

Deve-se observar que os totais gerados por classe e por período convergem para os valores definidos em $R_{c,p}$. Além disso, o total de eventos gerados por rodada, 3.417,2 com o HW (IC em 95%) igual a 3,68, também, converge para o valor $\sum_{c=1}^{C}\sum_{p=1}^{P} R_{c,p} = 3.420$.

[5] Todas as simulações foram feitas em um computadador com processador AMD Athlon 64X2 Dual Core Processor 4000+, 1GB RAM, com *Microsoft Windows XP Professional* e o *Arena Simulation*, versão 11. A programação foi feita com Microsoft *VisualBasic* 6.3, disponível no *Arena*.

[6] O tempo computacional foi de 0,7s por rodada.

Ainda para fins de exemplificação e validação do algoritmo, extraiu-se do vetor $T_{,}$ na última rodada da simulação, todos os eventos gerados para a classe 2 (‘*suporte’*) e período 3 (intervalo [60,90)), os quais são exibidos no Quadro 3.4. Para esta rodada da simulação, foram geradas exatamente 10 entidades. Note, no entanto, que esta igualdade nem sempre ocorre em virtude do processo estacionário definido por Φ. Conforme pode ser visto no Quadro 3.3, entre as 1000 rodadas da simulação, a menor quantidade de eventos gerados foi 2 e no máximo gerou-se 24 eventos para esta classe e período.

Tempos de chegada Eventos c=2 p=3
60,00
61,16
65,09
67,28
69,77
70,67
72,68
74,29
74,65
77,77

Quadro 3.4 - Tempos de chegadas para os eventos da classe 2 e período 3

3.1.5 *Implementação Alternativa no Arena Simulation Usando Exclusivamente Módulos ‘Modules’*

Com o intuito de mostrar as dificuldades de implementação nos ambientes de simulação, usando somente a infraestrutura disponível, a Figura 3.5 exibe a implementação do algoritmo proposto usando exclusivamente módulos (*modules*) do *Arena.*

O módulo ‘*Create 1’* gera uma única entidade que é encaminhada para o módulo ‘*ReadWrite 1*’, cuja finalidade é fazer a leitura dos dados de entrada que estão armazenados em uma planilha do *Microsoft Excel*.

O módulo ‘*Create 2*’ também gera uma única entidade que, em seguida, é separada em mais ‘*C – 1*’ entidades. Ou seja, a partir do módulo ‘*Separate 1*’ haverá tantas entidades quantas forem as classes ‘*C*’ existentes.

O módulo seguinte, '*Assign 1*', atribui à variável interna do modelo '*interv*' o valor 1, indicando que, neste momento, a simulação está no primeiro intervalo. O módulo '*Decide 1*' verifica se o tempo atual da simulação (variável interna '*tnow*' do *Arena*) ainda é menor ou igual ao período atual. Caso a variável '*tnow*' seja maior que o período atual, a entidade é encaminhada para o módulo '*Decide 2*', que verifica se todos os períodos já foram percorridos pela simulação. Caso afirmativo, encerra-se a simulação com o módulo '*Dispose 2'.* Caso contrário, o módulo '*Assign 2*' incrementa a variável '*interv*' em uma unidade, passando para o período seguinte.

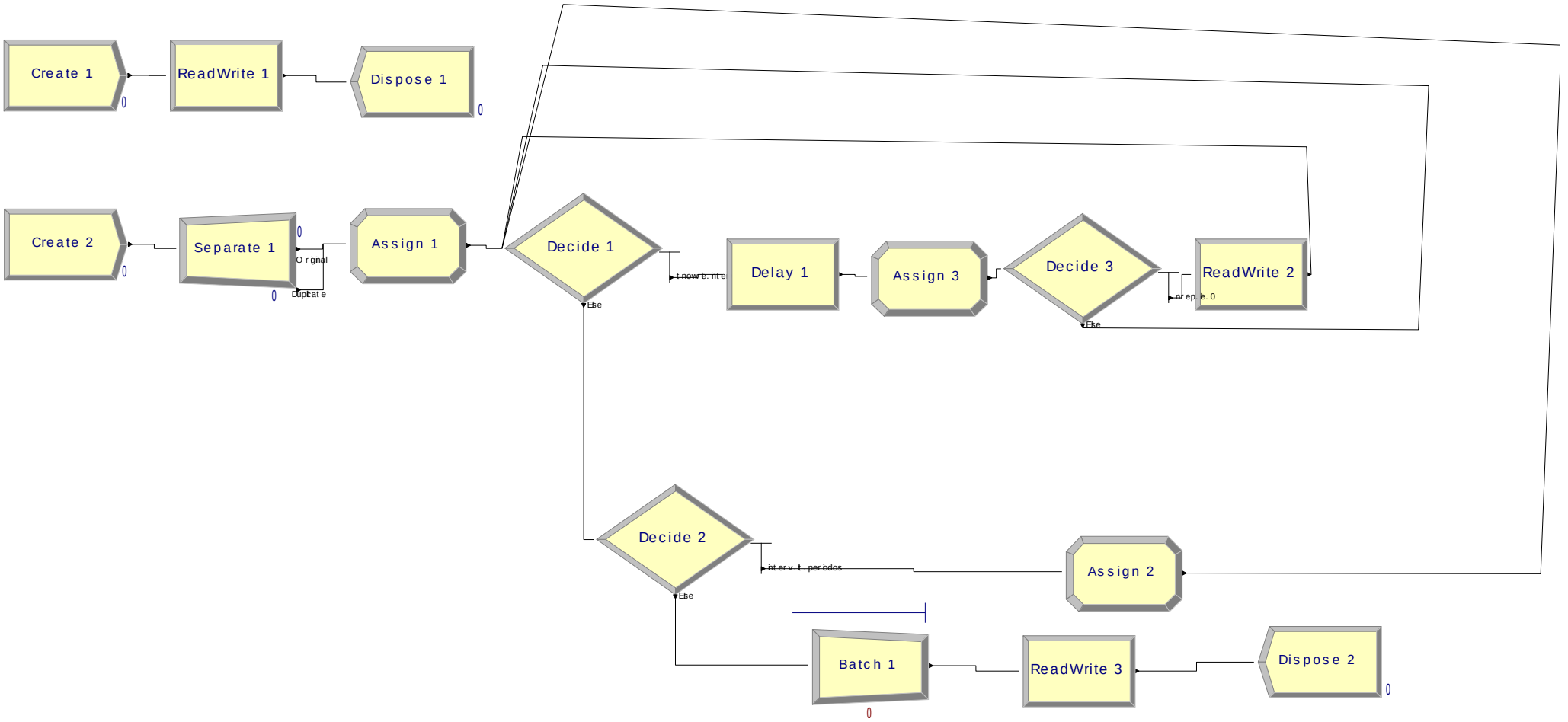

Figura 3.5 – Implementação do Algoritmo '*Principal* PR com taxas constantes por período, de uma origem para múltiplos destinos' no Arena usando exclusivamente módulos

Caso o teste do módulo '*Decide 1*' indique que a variável '*tnow*' ainda esteja dentro do intervalo do período atual, a entidade é encaminhada para o módulo '*Delay 1*' que retém a entidade pelo tempo igual a $\frac{expo(intervalo(interv+1) - intervalo(interv))}{RCP(classe,interv)}$.

A função interna do *Arena* '*expo(mean)'* gera um número distribuído exponencialmente com média igual ao parâmetro fornecido. Como o parâmetro passado é a diferença entre os intervalos, isto implica uma taxa da distribuição exponencial igual a 1. '*RCP(classe, interv)*' é a variável que armazena a taxa para a respectiva '*classe'* dentro de um intervalo '*interv*' definido. Esta operação é equivalente ao passo 17 do algoritmo '*DirectAlgorithmporClasse*' exibido no Quadro 3.2.

Os demais módulos realizam a contagem dos eventos gerados pela simulação, não carecendo de maiores explicações para o entendimento do trabalho. Apesar de essa implementação produzir os mesmos resultados que a solução em VBA, ela apresenta dois problemas fundamentais, quando comparada com a solução em VBA:

- Falta de legibilidade da solução que, em consequência, pode apresentar dificuldades de manutenibilidade;

- O preenchimento dos parâmetros do módulo *'ReadWrite 1'* , ver Figura 3.6, é feito individualmente para cada variável do problema. Isto significa que, se o tamanho do problema crescer, caberá ao projetista a alteração deste módulo, preenchendo as demais variáveis. Ou seja, a solução, além de tornar mais difícil a manutenção futura, não possui escalabilidade, tornando inviável sua utilização para problemas maiores.

Apesar destas desvantagens, a solução apresenta uma propriedade interessante. Como a geração dos eventos é feita de maneira concorrente para todas as classes, a execução do passo 10 (classificação dos eventos por ordem de chegada para todas as classes) do Algoritmo Principal (Quadro 3.1) não é necessária.

Recomenda-se, então, que os projetistas tenham em mente as questões de legibilidade e escalabilidade quando da implementação do algoritmo proposto nos ambientes computacionais em que serão utilizados.

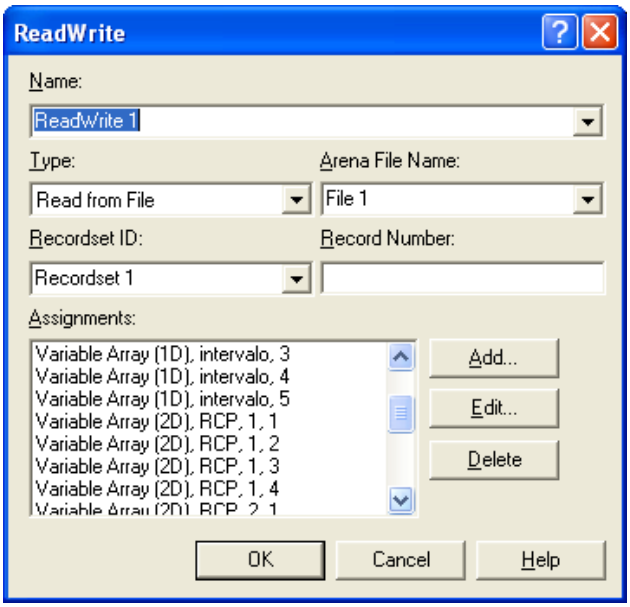

Figura 3.6 – Módulo '*ReadWrite 1*'

3.2 PROCESSOS DE RENOVAÇÃO COM TAXA CONSTANTE POR PERÍODO, DE MÚLTIPLAS ORIGENS PARA MÚLTIPLOS DESTINOS

São processos cuja taxa de chegada é constante por período e as entidades são originadas de múltiplas fontes (ou origens) para múltiplos destinos. Diversos exemplos poderiam ser citados para processos desta natureza. Abaixo, consideram-se dois exemplos:

1) uma central de atendimento ao público, na qual as requisições por demandas são originadas por vários meios (telefone, e-mail, correspondência, pessoalmente, etc.) e são classificadas ou destinadas internamente de acordo com o tipo de chamada;

2) modelagem de uma rede de computadores. Nestes sistemas as mensagens ou pacotes são originados de múltiplas origens para múltiplos destinos;

Considerando a similaridade com o método descrito anteriormente, no que se refere à implementação no *Arena Simulation*, o texto a seguir ater-se-á, exclusivamente, à exposição do algoritmo proposto para este tipo de processo não-estacionário.

3.2.1 Definindo as Variáveis do Problema

Seja um processo não-estacionário para o qual se tenha os seguintes dados de entrada:

- ***P***: quantidade de períodos ou faixas na simulação;
- ***C***: quantidade de classes definida para o processo na simulação;
- NC_c: indica o nome da classe *c*;
- $R_{clo,cld,p}$: taxa de chegada na simulação da classe de origem ´*clo*´ para a classe de destino ´*cld*´ no período *p*;
- B_p: valor temporal na simulação para cada período *p*, sendo $B_0 = 0$;
- Φ_1: inversa da função cumulativa (e seus parâmetros) do processo a modelar. Valem as observações do algoritmo anterior;
- Φ_2: inversa da função cumulativa (e seus parâmetros) que estabelece o valor do atributo *Capacidade* da entidade a ser gerada;

Com estes dados de entrada, deseja-se obter um vetor T, T_i, $i = 1...n$, sendo que T_i representa o *i*-ésimo evento a ser gerado na simulação.

Para a solução deste problema, também foi utilizada uma versão modificada do algoritmo *Direct Algorithm*, adaptando-o no que necessário para a geração de eventos de múltiplas origens para múltiplos destinos. Desta forma, fez-se uso, ainda, das seguintes variáveis:

- $C_{clo,cld,p}$: soma das taxas de chegada de uma determinada classe de origem '*clo'* para uma classe de destino '*cld'* de todos os períodos até *p* - $C_{clo,cld,p} = \sum_{i=1}^{p} R_{clo,cld,i}$;
- A_i: $A_i = \Phi$ e $S_i = S_{i-1} + A_i$. Então, $S_1, S_2,...S_n$. são os tempos de chegada dos eventos do processo estacionário definido por Φ;

- $\boldsymbol{S_n}$: tempos dos eventos para o processo estacionário definido pela função Φ, sendo $S_o = 0$ e $S_n = \sum_{i=1}^{n} A_i$;

- $\boldsymbol{T_{clo,cld,i}}$**:** tempos de chegada em ordem crescente para o processo não-estacionário da classe de origem '*clo'* para a classe de destino '*cld'*, onde *i* varia de *1* a *n* .

3.2.2 Algoritmo de Geração dos Eventos

No (Quadro 3.5) é apresentado o algoritmo principal que gera o vetor *T* cujo conteúdo são todos os eventos do processo não-estacionário, de acordo com as taxas definidas por período e por classe de origem e de destino em $R_{clo,cld,p}$.

```
Início Algoritmo Principal
1     Obtenha os dados de entrada (P, C, NC_c, R_clo,cld,p, B_p e Φ)
2     Para clo = 1 até C faça
3         Para cld = 1 até C faça
4             Para p = 1 até P faça
5                 C_clo,cld,p = C_clo,cld,p-1 + R_clo,cld,p
6             Fim para
7         Fim para
8     Fim Para
9     Para clo = 1 até C faça
10        Para cld = 1 até C faça
11            Chame DirectAlgorithmporClasse(T_clo,cld)
12    Fim para
13    Classifique, por ordem de chegada, todos os vetores T_clo,cld´s e mescle-os em um
      único vetor T no qual cada entrada T_i tenha o formato: tempo de chegada,
      Capacidade, clo, cld, NC_clo, NC_cld
14    Retorne T
Fim
```

Quadro 3.5 – Algoritmo *Principal* para PR com taxa constante por período, de múltiplas origens para múltiplos destinos

Depois de obter, na linha 1, todos os dados de entrada do problema, o algoritmo principal totaliza em $C_{clo,cld,p}$, na linha 5, para cada classe de origem, de destino e por período, os valores das taxas de chegada $R_{clo,cld,p}$. Na linha 11, o algoritmo chama, para cada classe de origem e de destino, o procedimento *DirectAlgorithmporClasse*, o qual constrói o vetor $T_{clo,cld}$ cujo conteúdo é o conjunto de eventos gerados da classe *clo* para a classe *cld*. Na linha 13,

todos os vetores $T_{clo,cld}$´s são mesclados em um único vetor *T*, que é o resultado esperado do algoritmo.

Acredita-se que não é necessária uma explicação mais ampla sobre o procedimento *DirectAlgorithmporClasse*, apresentado no Quadro 3.6, visto que é idêntico ao utilizado no item 3.1.2. Apenas, ressalte-se a adaptação feita nas variáveis $C_{clo,cld,p}$ e $T_{clo,cld,n}$ transformando-as em vetores de uma classe de origem *´clo´* para uma classe de destino *´cld´*. Outra diferença é que nas linhas 18 e 19 é atribuído um valor à variável *Capacidade* de acordo com a função Φ_2.

```
Início DirectAlgorithmporClasse (T_clo,cld)
1       Dimensione o vetor T_clo,cld,n com o tamanho C_clo,cld,P
2       n = 0
3       p = 1
4       S_n = 0
5       Faça
6               n = n + 1
7               Gere um número aleatório u~U[0,1]
8               A = Φ_1 (u)
9               S_n = S_n + A
10              Se S_n < C_clo,cld,P faça
11                      Enquanto (S_n – C_clo,cld,p) > 0 faça
12                              p = p + 1
13                      Fim enquanto
14                      Se n > C_clo,cld,P faça
15                              Redimensione o vetor T_clo,cld,n para o tamanho n
16                      Fim se
17                      T_clo,cld,n = B_p-1 + ((S_n - C_clo,cld,p-1) / R_clo,cld,p ) * (B_p - B_p-1)
18                      Gere um número aleatório u~U[0,1]
19                      Capacidade_n = Φ_2 (u)
20              Fim se
21      Fim faça enquanto S_n < C_clo,cld,P
22      Retorne T_clo,cld
Fim
```

Quadro 3.6 - Algoritmo DirectAlgorithm*porClasse* para PR com taxas constantes por período, de múltiplas origens para múltiplos destinos

3.2.3 Resultados do Algoritmo Proposto

Abaixo, apresenta-se um exemplo numérico com os seguintes dados de entrada, considerando uma rede de computadores composta por 3 servidores:

- $\boldsymbol{P} = 3$;
- $\boldsymbol{C} = 3$;
- $\boldsymbol{NC_c}$ = { ‘Serv1’, ‘Serv2’, ‘Serv3’}
- $\boldsymbol{B_p} = \{0, 30, 60, 90)$;
- $\boldsymbol{\Phi_1} = -ln(u)$;
- $\boldsymbol{\Phi_2} = 50 + (-ln(u)/\lambda)$, onde $\lambda = 0.02$. Neste caso, o atributo *Capacidade* da entidade gerada terá seus valores estabelecidos de acordo com uma distribuição exponencial de taxa 0.02. Portanto, o valor esperado do atributo *Capacidade* será igual a $E[\boldsymbol{\Phi_2}] = E[50 + (-ln(u)/\lambda)], = E[50] + E[(-ln(u)/\lambda)] = 50 + 1/\lambda = 100$;
- ***Rclo,cld,p =***

		Períodos p								
		1			2			3		
Classes clo	cld	1	2	3	1	2	3	1	2	3
1		0	20	30	0	200	300	0	20	30
2		40	0	60	400	0	600	40	0	60
3		70	80	0	700	800	0	70	330	0

Para estes dados, tem-se então um processo NSPP. De acordo com $R_{clo,cld,p}$, deverão ocorrer, em média, 800 eventos originados da classe 3 para a classe 2 no período 2, com intervalos entre chegadas distribuídos exponencialmente, dados pela função Φ**.**

Executou-se uma simulação com 1000 rodadas[7] e os seguintes resultados foram obtidos, conforme ilustra o Quadro 3.7.

[7] O tempo computacional foi de 0,86s por rodada.

Totais por classe e período						
Período	clo	cld	Média	HW	Mínimo	Máximo
1	1	1	0,00	0,00	0,00	0,00
		2	19,94	0,27	9,00	35,00
		3	29,91	0,33	11,00	45,00
	2	1	40,18	0,39	21,00	63,00
		2	0,00	0,00	0,00	0,00
		3	59,77	0,49	38,00	91,00
	3	1	69,94	0,51	45,00	95,00
		2	80,09	0,55	53,00	117,00
		3	0,00	0,00	0,00	0,00
2	1	1	0,00	0,00	0,00	0,00
		2	200,57	0,89	162,00	253,00
		3	300,51	1,06	244,00	365,00
	2	1	400,24	1,22	336,00	460,00
		2	0,00	0,00	0,00	0,00
		3	599,13	1,54	527,00	679,00
	3	1	698,82	1,65	612,00	791,00
		2	801,08	1,74	707,00	894,00
		3	0,00	0,00	0,00	0,00
3	1	1	0,00	0,00	0,00	0,00
		2	20,16	0,28	8,00	36,00
		3	30,05	0,33	14,00	50,00
	2	1	40,32	0,40	22,00	63,00
		2	0,00	0,00	0,00	0,00
		3	60,00	0,49	37,00	91,00
	3	1	70,09	0,55	43,00	104,00
		2	329,95	1,09	278,00	385,00
		3	0,00	0,00	0,00	0,00
Total de Eventos por rodada						
			3851,70	3,95	3649,00	4046,00

Quadro 3.7 - Totais de eventos gerados de classe de origem para destino e por período com IC em 95% para 1000 rodadas

Deve-se observar que os totais gerados por classe e por período convergem para os valores definidos em $R_{clo,cld,p}$. Além disso, o total de eventos gerados em média por rodada, 3.851,70 com o HW (IC em 95%) igual a 3,95, também converge para o valor $\sum_{clo=1}^{C}\sum_{cld=1}^{C}\sum_{p=1}^{P} R_{clo,cld,p} = 3.850$.

Ainda para fins de exemplificação e validação do algoritmo, extraiu-se do vetor T, na última rodada da simulação, todos os eventos gerados com origem na classe 1 para a classe 2, no período 1 (intervalo [0,30]), os quais são exibidos no Quadro 3.8. Para esta rodada da simulação, foram geradas 17 entidades com os tempos de chegada conforme mostra o Quadro

3.8. Dentre as 1000 rodadas da simulação, a menor quantidade de eventos gerados foi 9 e no máximo gerou-se 35 eventos.

Tempos de chegada
0,726146
2,333614
2,970486
5,491671
6,957974
7,142878
7,450276
8,221402
8,736456
10,00661
10,34543
14,95051
16,24766
17,72764
17,76146
18,25244
30

Quadro 3.8 - Tempos de chegadas para os eventos com origem na classe 1 para a classe 2 dentro do período 1

3.2.4 *Implementação Alternativa no Arena Simulation Usando Exclusivamente 'Modules'*

A exemplo da Figura 3.5, a Figura 3.7 mostra a implementação do algoritmo *'Principal para PR com taxa constante por período, de múltiplas origens para múltiplos destinos'*. Essa implementação usa exclusivamente módulos do *Arena*. Como esta implementação é análoga à exibida no item 3.1.5, não é necessário maior detalhamento.

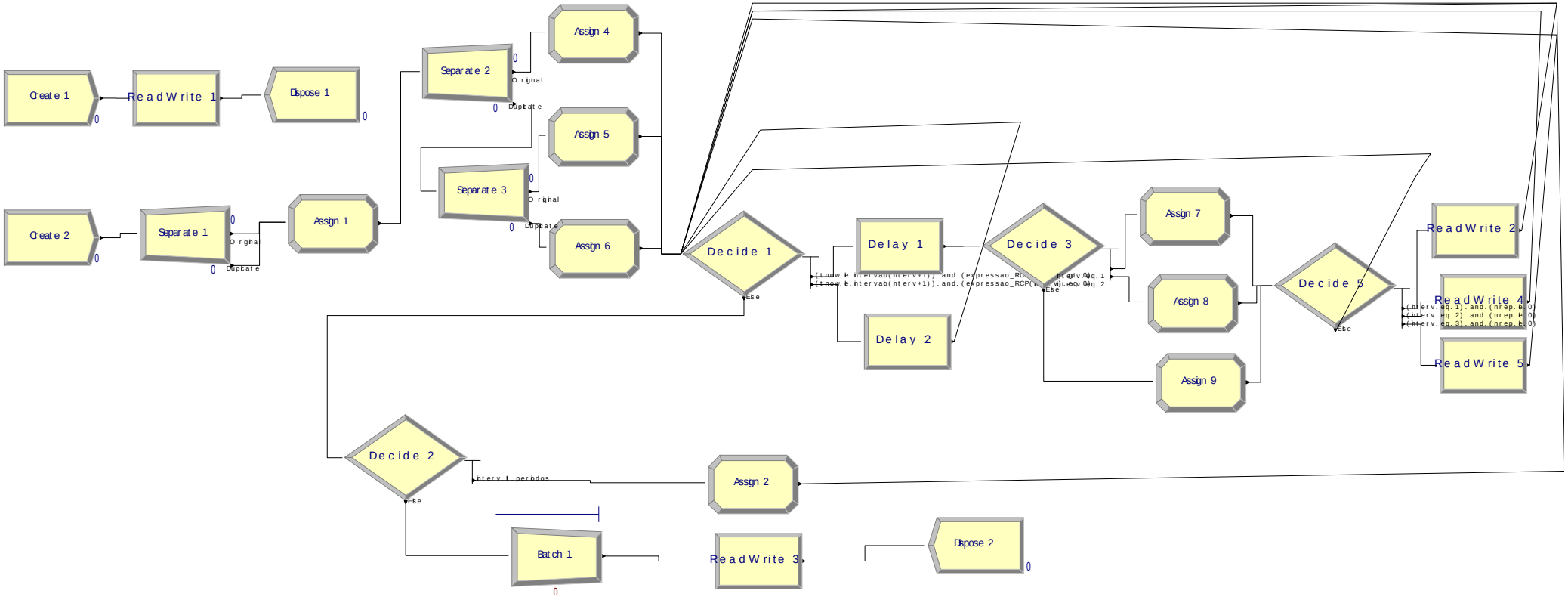

Figura 3.7 – Implementação do Algoritmo Proposto para Múltiplas Classes de Origens e Destinos Usando '*Modules'* no *Arena*

Essa implementação, mesmo produzindo os mesmos resultados que a solução em *VBA*, também apresenta dois problemas:

- Evidentemente, agravou-se a falta de legibilidade da solução em relação ao algoritmo proposto;

- O preenchimento do módulo *'ReadWrite 1'* (ver Figura 3.8), que faz a leitura dos dados de entrada do problema, é feito individualmente para cada variável do problema. Isto também agrava o problema de escalabilidade, principalmente, porque a variável *´RCP'* terá tamanho, neste caso, igual a C^2*P. Fica claro que é impraticável o uso desta solução para modelar uma rede de computadores com 100 nós de rede, com taxas distintas de pacotes entre os nós e subdivididos em, por exemplo, 20 períodos.

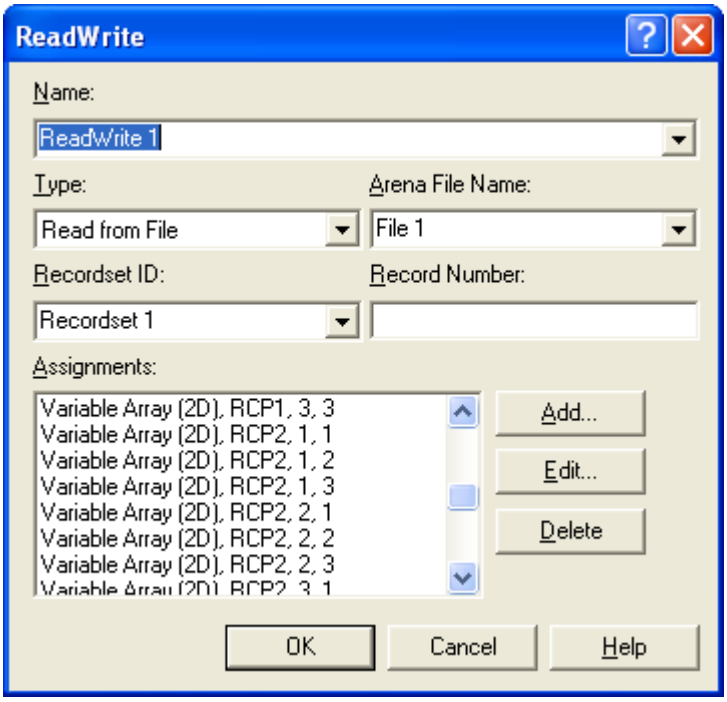

Figura 3.8 – Módulo '*ReadWrite 1'*

3.3 PROCESSOS DE RENOVAÇÃO COM TAXAS DEFINIDAS POR MEIO DE PLANEJAMENTO

A Figura 3.9 apresenta os componentes básicos do modelo para PRs não estacionários com taxas definidas por meio de planejamento, o qual se pretende resolver. Neste tipo de modelo, a geração ou a entrada das entidades ocorre diretamente vinculada a um destino específico. Além disso, a quantidade de entidades gerada para um determinado destino deve, ao final de uma rodada da simulação, estar plenamente de acordo com os valores pré-estabelecidos ou definidos no início da simulação. Isto significa que a quantidade de entidades gerada por período da simulação (horas, dias, semanas, etc.) será estabelecida em função de um planejamento necessário para o atendimento de demanda futura. Desta forma, a diferença principal em relação aos modelos anteriores é que, para este caso, as taxas por período não são pré-definidas, mas sim calculadas em função de uma demanda futura.

Cada destino possui uma unidade de armazenamento (*buffer*) que possui uma capacidade limitada, em unidades, de armazenamento (*CE*). Além disso, há uma limitação na capacidade de tratamento (*CT*), por período, no destino. Assim sendo, não se pode receber, por período, uma quantidade superior a *CT*, sob pena da ocorrência de filas no entorno do destino.

Para cada destino, há uma demanda futura por classes ($D_{d,c}$), em unidades, a ser atendida. Os períodos de demanda *´d´* variam de [1...*PD*] e a quantidade de classes *´c´* varia de [1...*C*].

As demandas $D_{d,c}$ são atendidas por meios de transporte (*MT*) que podem variar de [1...*QMT*]. Entende-se por meio de transporte qualquer mecanismo utilizado com a finalidade de transportar as unidades demandadas de suas origens até o destino. Cada *MT* possui uma determinada capacidade (CMT_{mt}), *´mt´* = 1...*QMT*, em unidades. Desta forma, cada *MT* gerado no sistema "leva" ao destino *CMTmt* unidades para que ao final da programação a demanda necessária seja atendida.

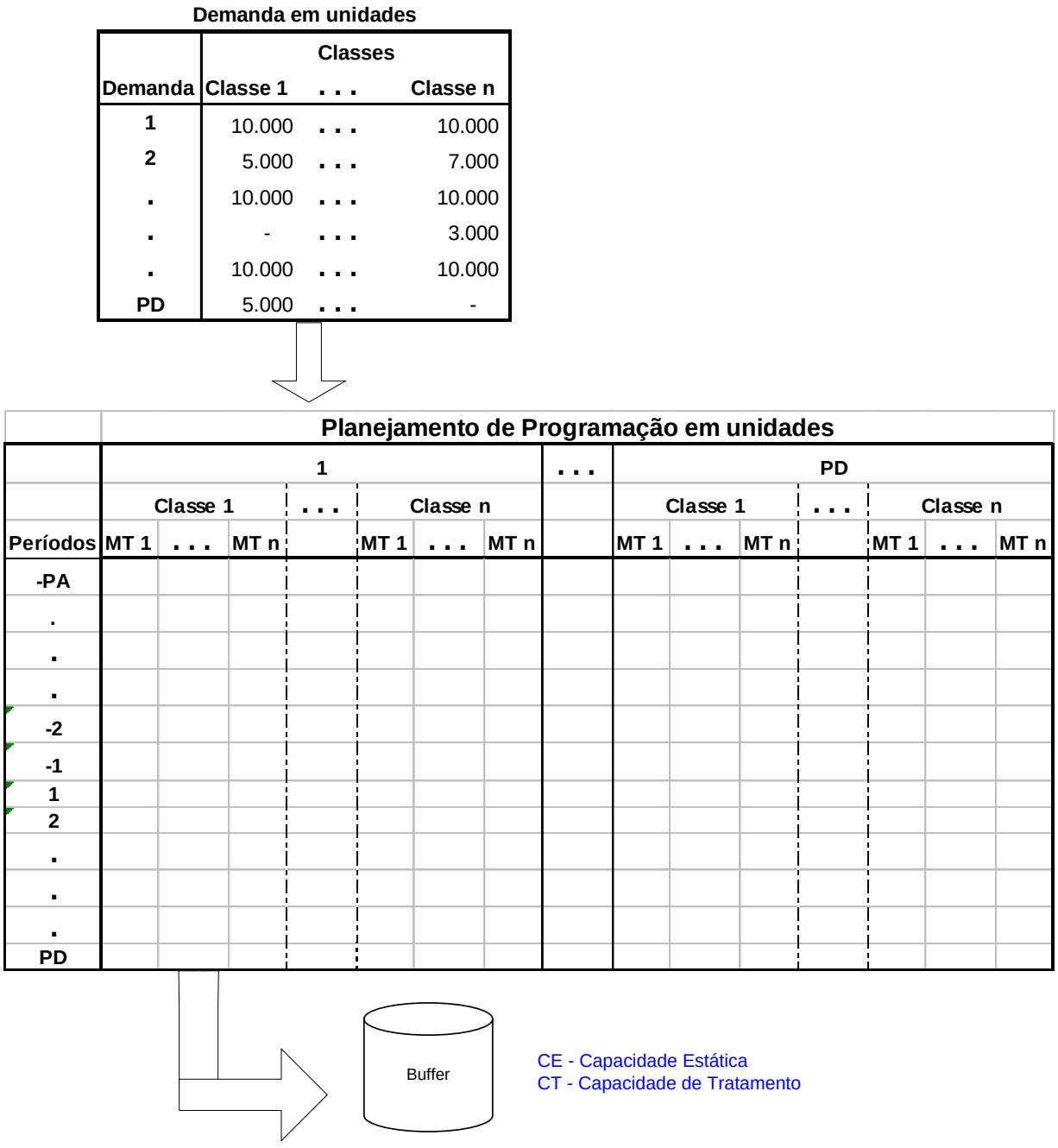

Figura 3.9 - Componentes básicos do modelo de simulação para processos não estacionários com taxas calculadas

É possível que, antes de se iniciarem os períodos de demanda, ainda haja uma folga, também em períodos, para o envio das unidades ao destino. Esta folga é definida no modelo como períodos anteriores (*PA*) aos períodos de demanda (*PD*). Em algumas aplicações este período de folga pode ser visto e, portanto, configurado como o período de aquecimento da simulação (*Warm-up*), conforme será visto na aplicação do modelo. Então, os períodos de

planejamento de programação variam de [-*PA*,...,-1,1,...,*PD*]. Os valores ($VTMT_{p,d,c,mt}$) estabelecem as quantidade em unidades de meios de transportes que são gerados a fim de se atender à demanda futura, sendo que *p* = -*PA*,...,-1, ,1...*PD*, *d*=1...*PD*, *c* = 1...*C* e *mt* = 1...*MT*.

3.3.1 Definindo as Variáveis do Problema

Seja um PR não-estacionário para o qual se tenha os seguintes dados de entrada para um determinado destino:

- ***CE***: capacidade, em unidades de armazenamento, do *buffer* no destino;
- ***CT***: capacidade de tratamento ou recepção de armazenamento no destino, em unidades, por período;
- ***PS***: quantidade de períodos de segurança a ser adotado. Considerando uma demanda de uma determinada classe *c*, em um período futuro *d* , *PS* indica que esta demanda deve ser atendida, preferencialmente, até o período (*d* – *PS*-1);
- ***PD***: quantidade de períodos de demanda. São os períodos para os quais haverá programação de demanda;
- ***PA***: quantidade de períodos disponíveis anteriores ao início dos períodos de demanda;
- ***LSMin***: quando houver programação para um período, *LSMin* indica o tamanho do lote mínimo, em unidades, que pode ser gerado;
- ***LSMax***: quando houver programação para o período, *LSMax* indica o tamanho do lote máximo, em unidades, que pode ser gerado;
- ***C***: quantidade de classes para o destino;
- $\boldsymbol{NC_c}$: indica o nome da classe *c*;
- ***QMT***: quantidade de meios de transporte disponíveis para levar as unidades ao destino;

- $\boldsymbol{NMT_{mt}}$: nome do meio de transporte *mt*;

- $\boldsymbol{CMT_{mt}}$: capacidade, em unidades, do meio de transporte *mt*;

- $\boldsymbol{PMT_{c,mt}}$: indica qual é o percentual da demanda futura para a classe *c* que será atendida pelo meio de transporte *MT*;

- $\boldsymbol{D_{d,c}}$: define qual é a demanda, em unidades, no período (d = 1...*PD*) para a classe *c*;

- **Φ**: inversa da função cumulativa (função de distribuição) do processo a modelar. Esta função define o ajuste (distribuição) que será feito nos intervalos de chegada, dentro de cada período *p*, para o conjunto de valores calculados pelo algoritmo.

Para a solução do problema, definem-se as seguintes variáveis de decisão:

- $\boldsymbol{VT_{p,d,c}}$: são os valores necessários, em unidades, por período (p = -*PA*,...,-1, ,1...*PD*, d=1...*PD* e c = 1...*C*), a fim de que a demanda $D_{d,c}$ seja atendida;

- $\boldsymbol{VTMT_{p,d,c,mt}}$: são os valores necessários, em unidades de meios de transportes, por período (p = -*PA*,...,-1, ,1...*PD*, d=1...*PD*, c = 1...*C* e mt = 1...*QMT*), a fim de que a demanda $D_{d,c}$ seja atendida. Neste ponto, é importante destacar que esta variável é similar à variável ***R*** dos métodos anteriores, a qual definia a taxa de chegada para o PR não-estacionário. No método em questão, *VTMT* é a taxa de chegada para o PR não-estacionário sendo, entretanto, calculada em função de uma demanda futura e não pré-estabelecida.

Inicialmente, por meio dos dados de entrada, os valores de $VT_{p,d,c}$, em unidades, são calculados levando em consideração: as demandas futuras que devem ser atendidas preferencialmente até o período (d–*PS*-1), as capacidades de tratamento por período nos destinos e os tamanhos de lotes mínimos e máximos. $VTMT_{p,d,c,mt}$ é a conversão dos valores $VT_{p,d,c}$ (que estão em unidades de demanda) para unidades de meios de transporte.

Considerando o modelo proposto na Figura 3.9, os seguintes objetivos gerais devem ser atingidos pelo método numérico a ser desenvolvido:

1) Durante uma rodada da simulação, a quantidade demandada $D_{d,c}$ deve ser levada aos destinos pelos meios de transporte, por meio de uma programação que satisfaça às seguintes condições:

   i. $D_{d,c} = \sum_{p=-PA}^{PD} VT_{p,d,c}$, para todo d = 1...PD e c = 1...C. Esta condição garante que o total programado, em unidades, é igual à demanda no período d para a classe c;

   ii. $\sum_{p=-PA}^{PD} VT_{p,d,c} \leq \sum_{p=-PA}^{PD} \sum_{mt=1}^{MT} VTMT_{p,d,c,mt} * CMT_{mt}$, de forma que a retirada de apenas uma unidade do meio de transporte na programação, com a menor CMT, já seja suficiente para que esta condição não seja atendida. As duas condições *i* e *ii* garantem, simultaneamente, que as quantidades transportadas para o destino atenderão à demanda utilizando a menor quantidade de meios de transporte possível;

   iii. $\frac{\sum_{p=-PA}^{PD} VTMT_{p,d,c,MTe} * CMT_{MTe}}{\sum_{p=-PA}^{PD} \sum_{mt=1}^{MT} VTMT_{p,d,c,mt} * CMT_{mt}} \cong PMT_{c,MTe}$, para todas as demandas d = 1...PD, c = 1...C e MT_e = 1...QMT. Esta condição garante que os totais transportados por cada meio de transporte estarão, o mais próximo possível, de acordo com os percentuais definidos em $PMT_{c,mt}$. Note que a expressão indica uma aproximação em percentual dos valores, já que o somatório das CMT_{mt} pode não ser múltiplo do total da demanda $D_{d,c}$;

   iv. $LSMin \leq VT_{p,d,c} \leq LSMax$, para todo período p de uma demanda d de uma classe c. Esta condição impõe que o total a ser transportado, em um determinado período, esteja entre os valores de lote mínimo e máximo.

2) A programação deve ser definida de forma que, <u>preferencialmente</u>:

- $D_{d,c} = \sum_{p=-PA}^{n} VT_{p,d,c}$, n = -*PA*...-1,1...*PD* e $n \leq$ (*d*-*PS*-1). Esta condição garante que as unidades necessárias já tenham sido programadas, preferencialmente, até *PS* períodos anteriores à data da demanda *d*;

3) Ainda que os *buffers* não estejam completamente cheios, a quantidade de unidades, por período da simulação, geradas e enviadas aos destinos, não pode ser, preferencialmente, superior à sua capacidade de recepção e tratamento, pois filas indesejáveis surgirão no entorno do destino durante a rodada da simulação. Então:

- $\sum_{d=1}^{PD}\sum_{c=1}^{C} VT_{p,d,c} \leq CT$, para todo p = -*PA*,...-1,1,...*PD*;

Com os dados de entrada e os objetivos definidos acima, deseja-se obter um vetor T_i, i = 1...n , sendo que T_i representa o *i*-ésimo evento a ser gerado na simulação. Para cada evento, T_i indica: o momento temporal da chegada, o meio de transporte utilizado, a classe e à qual demanda futura se referem as unidades transportadas.

3.3.2 Problema do Planejamento da Programação

Para o problema definido anteriormente, supondo que se disponha de *n* períodos para o planejamento de uma demanda, onde n é igual a (*PA* + *PD*), e que esta demanda será totalmente atendida em um único período, então há C_n^1 possibilidades para o planejamento. Supondo que o planejamento seja feito em 2 períodos, não necessariamente consecutivos, então há C_n^2 possibilidades. Continuando o raciocínio, o total de possibilidades para o planejamento da programação de uma única demanda é dado por $\sum_{i=1}^{n} C_n^i$. Como no método proposto há *PDs* demandas a serem atendidas, o total de possibilidades é dado por $\left(\sum_{i=1}^{n} C_n^i\right)^{DP}$. Evidentemente que este é um problema combinatorial e que apresenta um crescimento exponencial à medida que crescem os períodos de demanda. Portanto, dependendo dos valores de *PA* e *PD*, o crescimento torna-se explosivo.

Exigindo-se que a programação de uma demanda seja feita de forma consecutiva, o que é bastante razoável para a maioria dos problemas reais, o total de possibilidades pode ser reduzido substancialmente. Supondo que a programação de uma demanda se inicie no período 1, então, o seu término pode se dar dentro do próprio período 1 até *n*, com o total de *n* possibilidades. Caso o início da programação se dê no período 2, então o seu término pode se dar dentro do próprio período 2 até n, com o total de (n – 1) possibilidades. Prosseguindo com o mesmo raciocínio, tem-se, então n+(n-1)+(n-2)+...+1 possibilidades, sendo que a soma desta progressão aritmética é igual a $\left(\frac{(1+n)n}{2}\right)$. Considerando que o objetivo é o de realizar o planejamento de *PDs* períodos de demanda, então, o total de possibilidades é de $\left(\frac{(1+n)n}{2}\right)^{PD}$, com n = (*PA* + *PD*). Apesar da redução significativa, é evidente que, ainda assim, o problema cresce exponencialmente à medida que cresce a quantidade de períodos de demanda *PD*.

Com o intuito de reduzir o espaço de busca da solução, adotou-se outra medida a fim de limitar o crescimento exponencial do problema. Fixou-se o período τ como um período obrigatório no planejamento, sendo $\tau = d - PS - 1$, d = 1...*PD*, onde *d* é o período da demanda que está sendo planejada. Ou seja, $VT_{\tau,d,c} \neq 0$. Além disso, manteve-se a exigência de a programação ser feita em períodos consecutivos, não havendo interrupção após a programação ter sido iniciada.

A Figura 3.10 mostra o relacionamento entre τ, *d*, *PA* e *PD*. Supondo que a programação se encerre no período τ, portanto, não avançando no período de segurança, há (*PA* + *d* – *PS* – 1) possibilidades para o planejamento da demanda. Caso seja necessário, para o planejamento de uma demanda d, avançar dentro do período de segurança ou mesmo depois da data de demanda, então, restam ainda (*PD* – *d* + *PS* + 1) possibilidades. A soma destas duas expressões resulta em (*PA* + *PD*) possibilidades para cada planejamento de demanda *d*. Desta forma, o total de possibilidades para o planejamento de todas as demandas é de $(PA+PD)^{PD}$.

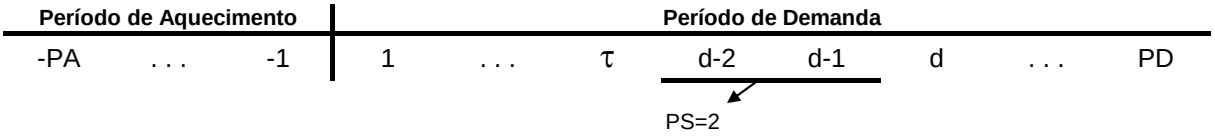

Figura 3.10 – Relacionamento entre as variáveis τ, d, PA e PD

Para que se tenha noção da redução da dimensão do problema, sendo $PA = 5$ e $PD = 20$ e, então, $n = 25$, e planejamento para apenas uma classe c, tem-se que: $\left(\sum_{i=1}^{n} C_n^i\right)^{PD}$ = 3,273E+150, $\left(\frac{(1+n)n}{2}\right)^{PD}$ = 1,728E+50 e $(PA+PD)^{PD}$ = 9,095E+27.

3.3.3 Algoritmo de Geração dos Eventos

No Quadro 3.9 é apresentado o algoritmo principal que gera o vetor T cujo conteúdo são todos os eventos, do processo PR não-estacionário, necessários para que as demandas $D_{d,c}$ sejam atendidas.

```
Início Algoritmo Principal
1     Para destino = 1 até todosOsDestinos
2         Obtenha os dados de entrada
3         Estabeleça o Planejamento da Programação, calculando os valores VT_{p,d,c}
4         Converta VT_{p,d,c} para VTMT_{p,d,c,mt}
5         Para d = 1 até PD faça
6             Para c = 1 até C faça
7                 Para mt = 1 até QMT faça
8                     Chame DirectAlgorithmTaxasCalculadas(T_{d,c,mt}, d, c, mt)
9                 Fim para
10            Fim para
11    Fim para
12    Classifique, por ordem de chegada, todos os vetores T_{d,c,mt} e mescle-os em um único
      vetor T_{destino} no qual cada entrada tenha o formato: tempo de chegada,PD, c ,mt.
13    Mesclar, por ordem de chegada, todos os vetores T_{destino} em um único vertor T, o qual
      tem o formato : destino, tempo de chegada, PD, c, mt.
14    Retorne T
Fim
```

Quadro 3.9 - Algoritmo *Principal* processos de renovação com taxas definidas por meio de planejamento

Para cada destino que se necessite realizar a geração de entidades, obtém-se, na linha 2, todos os dados de entrada do destino. Em seguida, o algoritmo *Principal* calcula os valores de $VT_{p,d,c}$ a fim de que as demandas $D_{d,c}$ sejam atendidas, de acordo com os objetivos definidos em 3.3.1. Na linha 4, os valores de $VT_{p,d,c}$, que estão em unidades de demanda, são convertidos para $VTMT_{p,d,c,mt}$, que nada mais são do que as taxas de chegada dos meios de transporte para o modelo de simulação. Em seguida, para todas as demandas *d*, de todas as classes *c* e para todos os meios de transporte *mt*, o procedimento *DirectAlgorithmTaxasCalculadas* é chamado para construir o vetor $T_{d,c,mt}$ cujo conteúdo serão os tempos de chegada, de um destino específico, na simulação para a demanda *d* da classe *c* usando o meio de transporte *mt*. Na linha 12, todos os vetores $T_{d,c,mt}$ são classificados e mesclados em um único vetor $T_{destino}$. Na linha 13, todos os vetores $T_{destino}$ são mesclados em um único vetor *T*, que é o resultado final do algoritmo.

3.3.3.1 Estabelecendo o Planejamento da Programação

Considerando os dados de entrada e as variáveis de decisão já discutidas, apresenta-se o problema de programação linear inteira mista (PPLIM), que realiza o planejamento da programação. Os resultados apresentados pelo PPLIM foram satisfatórios, o que não significa que outra modelagem ou técnica não possa ser usada, substituindo o passo 3 do algoritmo *Principal*. O modelo PPLIM usado foi:

Minimizar $Z = p1*ECT + p2*EPA + p3*QPS + p4*QPD$ (3.1)

Onde:

Cálculo *ECT* – Excesso Capacidade de Tratamento, quanto trouxe além da capacidade de tratamento por período estabelecido por CT

$$ELp = \sum_{d=1}^{PD} \sum_{c=1}^{C} VT_{p,d,c} \quad ,\forall\ p = 1...(PA + PD) \tag{3.2}$$

$$ELp - CT - ELCTP_p + ELCTN_p = 0 \quad ,\forall\ p = 1...(PA + PD) \tag{3.3}$$

$$ECT = \sum_{p=1}^{PA+PD} ELCTP_p \tag{3.4}$$

Cálculo *EPA* – Excesso Período de Aquecimento, quanto trouxe além da Capacidade Estática dentro do período de aquecimento

---

$$\sum_{p=1}^{PA} EL_p - CE - EPA + APA = 0 \qquad (3.5)$$

Cálculo *QPS* – Quantidade no Período de Segurança, quanto trouxe dentro do período de segurança

---

$$LDS_{d,c} = \sum_{p=d+PA-PS}^{d+PA-1} VT_{p,d,c} \qquad ,\forall\ d = 1...PD \text{ e } \forall\ c = 1...C \qquad (3.6)$$

$$QPS = \sum_{d=1}^{PD}\sum_{c=1}^{C} LDS_{d,c} \qquad (3.7)$$

Cálculo *QPD* – Quantidade Depois da Demanda, quanto trouxe depois do período previsto para a demanda

---

$$LDD_{d,c} = \sum_{p=d+PA}^{PA+PD} VT_{p,d,c} \qquad ,\forall\ d = 1...PD \text{ e } \forall\ c = 1...C \qquad (3.8)$$

$$QPD = \sum_{d=1}^{PD}\sum_{c=1}^{C} LDD_{d,c} \qquad (3.9)$$

Sujeito a:

Restrições para indicar em *DX* se há demanda no período *d* para a classe *c*

$$D_{d,c} \geq DX_{d,c} \qquad ,\forall\ d = 1...PD \text{ e } \forall\ c = 1...C \qquad (3.10)$$

$$D_{d,c} \leq BIG\_M * DX_{d,c} \quad ,\forall\ d = 1...PD \text{ e } \forall\ c = 1...C \tag{3.11}$$

Restrições para indicar em $X_{p,d,c}$ se $VT_{p,d,c}$<>0 ou não

---

$$VT_{p,d,c} \geq X_{p,d,c} \quad ,\forall\ p=1...PA+PD,\ \forall\ d=1...PD \text{ e } \forall\ c=1...C \tag{3.12}$$

$$VT_{p,d,c} \leq BIG\_M * X_{p,d,c} \quad ,\forall\ p=1...PA+PD,\ \forall\ d=1...PD \text{ e } \forall\ c=1...C \tag{3.13}$$

Restrições para deixar o envio contíguo até o período imediatamente anterior ao de segurança

---

$$X_{(PA-PS+d-1),d,c} = DX_{d,c} \quad ,\forall\ d = 1...PD \text{ e } \forall\ c = 1...C \tag{3.14}$$

$$VT_{p,d,c} \geq VT_{p-1,d,c} \quad ,\forall\ d = 1...PD,\ \forall\ c = 1...C \text{ e } \forall\ p = 2...PA+d-PS-1, \tag{3.15}$$

$$VT_{p,d,c} \geq VT_{p+1,d,c} \quad ,\forall\ d = 1...PD,\ \forall\ c = 1...C \text{ e } \forall\ p = PA+d-PS...PA+PD, \tag{3.16}$$

O total gerado em uma programação *d* deve ser igual ao total da demanda para aquele período

---

$$D_{d,c} = \sum_{p=1}^{PA+PD} VT_{p,d,c} \quad ,\forall\ d = 1...PD \text{ e } \forall\ c = 1...C \tag{3.17}$$

Restringem os valores máximos e mínimos para *VT*

$$VT_{p,d,c} \geq LSMin * X_{p,d,c} \quad ,\forall\ p=1...PA+PD,\ \forall\ d=1...PD \text{ e } \forall\ c=1...C \tag{3.18}$$

$$VT_{p,d,c} \leq LSMax * X_{p,d,c} \quad ,\forall\ p=1...PA+PD,\ \forall\ d=1...PD \text{ e } \forall\ c=1...C \tag{3.19}$$

Variáveis binárias

$$X_{p,d,c} = \begin{cases} 0 & se \quad VT_{p,d,c} = 0 \\ 1 & se \quad VT_{p,d,c} \neq 0 \end{cases} \tag{3.20}$$

$$,\forall\ d = 1...PD \text{ e } \forall\ c = 1...C$$

$$DX_{d,c} = \begin{Bmatrix} 0 & se & D_{d,c} = 0 \\ 1 & se & D_{d,c} \neq 0 \end{Bmatrix}$$

A Figura 3.11 apresenta um exemplo numérico que torna mais claro o relacionamento entre as variáveis do método proposto. Para este exemplo, considera-se um planejamento para um período de demanda igual a 5 e se dispõe de 14 períodos de aquecimento. Isto significa que ainda se dispõe de 14 períodos até que chegue o período da primeira demanda cuja necessidade é de 30.000 unidades. Além disso, foi estipulado que o período de segurança é igual a 2, sendo, então, desejável que todas as unidades tenham sido geradas com até 2 períodos de antecedência ao início da demanda.

A figura mostra, também, os valores de *VT* que serão gerados por período para cada demanda necessária. Observa-se que, para este caso, *QPS* e *QPD* são iguais à zero. Isto significa que não foi necessário avançar nem nos períodos de segurança e tampouco após o período previsto para o início de cada demanda. O total gerado dentro do período de aquecimento ($-14 \leq p \leq -1$) é igual a 106.000. Se for considerado, então, que o valor do dado de entrada *CE* é igual a 96.000, tem-se que *EPA* é igual a 10.000. Houve, portanto, um excesso de 10.000 unidades dentro do período de aquecimento. Para a maioria dos períodos considerados, o somatório máximo de *VT* para cada demanda foi de 8.216 unidades. Além disso, caso o valor do dado de entrada *CT* seja igual a 9.000, o valor de *ECT* é igual à zero, pois nunca se infringiu a capacidade por período de tratamento no destino. Por fim, o leitor deve observar que o somatório dos valores programados para cada demanda *d* atende às unidades requeridas.

	Período	Demanda	Quantidade por Período	VT 1	2	3	4	5
	-14		5.347	2500	705	2143		
	-13		5.347	2500	705	2143		
	-12		8.147	2500	705	2143		2800
	-11		8.147	2500	705	2143		2800
	-10		8.216	2500	773	2143		2800
	-9		8.216	2500	773	2143		2800
P	-8		8.216	2500	773	2143		2800
A	-7		8.216	2500	773	2143		2800
	-6		8.216	2500	773	2143		2800
	-5		8.216	2500	773	2143		2800
	-4		8.216	2500	773	2143		2800
	-3		8.216	2500	773	2143		2800
	-2		8.143		3000	2143		3000
	-1		5.143			2143		3000
	1	30.000				QPS		3000
	2	12.000						3000
P	3	30.000						
D	4	-			QPD			
	5	40.000						

Figura 3.11 – Relacionamento entre as variáveis *PA*, *PD*, *VT*, *QPS* e *QPD*

O objetivo do PPLIM apresentado anteriormente não é o de otimizar o planejamento da programação, mas sim apresentar alternativas viáveis para que, por meio da simulação, se possa avaliar a adequabilidade daquela solução. A função objetivo trabalha com um vetor de pesos, fornecidos pelo projetista da simulação, junto com os dados de entrada definidos em 3.3.1. É claro que, dependendo da demanda necessária e das restrições impostas ao problema, notadamente a capacidade estática (*CE*), a quantidade de períodos de segurança (*PS*) e a capacidade de tratamento por período (*CT*), é possível que não se consiga planejar as demandas de forma que todas estas restrições sejam atendidas simultaneamente. Desta forma, estas restrições foram relaxadas por meio de programação multiobjetivo (*goal programming*), conforme a expressão (3.1).

Para o cálculo do excesso havido no planejamento em relação à capacidade de tratamento no destino,(3.2) totaliza em EL_p o total dos valores gerados por período para cada demanda havida em cada classe. A expressão (3.3) atribui em $ELCTP_p$ o excesso que pode ter ocorrido em relação à capacidade de tratamento por período e, em contrapartida, atribui zero para $ELCTN_p$. Caso o valor planejado esteja abaixo da capacidade de tratamento, $ELCTP_p$ terá valor igual a zero e $ELCPN_p$ fica com a diferença entre estes valores. A equação (3.4) totaliza

em *ECT* os excessos que podem ter ocorrido em relação à capacidade de tratamento em todos os períodos.

Caso tenha havido excesso dos valores planejados, dentro do período de aquecimento, em relação à capacidade estática,(3.5) atribui à *EPA* este excesso e *APA* fica com zero. Caso os valores planejados estejam abaixo da capacidade estática, à *APA* é atribuída esta diferença, ficando *EPA* com zero.

Para o cálculo de *QPS*, a equação (3.6) totaliza, para cada demanda de uma classe, em $LDS_{d,c}$ os totais planejados dentro do período de segurança para cada demanda. Assim, *QPS* é o somatório de todos os valores de $LDS_{d,c}$.

$LDD_{d,c}$ totaliza, em (3.8), os valores planejados após o início previsto para uma demanda *d* de uma classe *c*. *QPD*, então, é o somatório de todos os valores planejados após o início previsto para uma demanda *d* de uma classe *c*.

O conjunto das restrições (3.10) e (3.11) faz com que a variável binária $DX_{d,c}$ assuma o valor 0 se não houver demanda no período *d* para a classe *c* e 1, caso contrário. O conjunto das restrições (3.12) e (3.13) faz com que a variável binária $X_{p,d,c}$ assuma o valor 0 se não houver programação no período *p* para a demanda *d* para a classe *c*.

A restrição (3.14) faz com que haja obrigatoriamente programação no período (*PA-PS+d*-1) imediatamente anterior ao início do período de segurança (ver item 3.3.2). Se $DX_{d,c}$ for igual a 1, também o será a variável $X_{(PA-PS+d-1),d,c}$. A restrição (3.15) impõe que, desde o período 2 até o período *PA+d-PS*-1, não haja um período com programação seguido de outro período com ausência de programação. Ou seja, impõe a contigüidade no planejamento. A mesma imposição é feita na restrição (3.16) nos períodos subsequentes, e inclusive, ao período de segurança.

A restrição (3.17) impõe que o total planejado para uma demanda *d* para uma classe *c* seja igual ao valor $D_{d,c}$. As restrições (3.18) e (3.19) especificam respectivamente os limites mínimos e máximos para cada período.

A restrição (3.20) impõe que as variáveis $X_{p,d,c}$ e $DX_{d,c}$ sejam binárias. Ressalte-se que não é necessário que variável de decisão $VT_{p,d,c}$ seja binária, pois estes são valores

intermediários no cálculo das quantidades $VTMT_{p,d,c,mt}$, que devem ser obrigatoriamente variáveis inteiras como se observará a seguir.

3.3.3.2 Convertendo o Planejamento da Programação $VT_{p,d,c}$ para $VTMT_{p,d,c,mt}$

Os valores calculados em $VT_{p,d,c}$ são dados em unidades de demanda. No entanto, as entidades que entram na simulação, de acordo com o método proposto, são os meios de transporte que possuem uma determinada capacidade de transporte dado por CMT_{mt}. Assim sendo, é necessário proceder à conversão dos valores calculados em $VT_{p,d,c}$ para unidades de meios de transporte dadas por $VTMT_{p,d,c,mt}$. O algoritmo do Quadro 3.10 mostra como esta conversão é realizada.

A fim de que a conversão de $VT_{p,d,c}$ para $VTMT_{p,d,c,mt}$ possa ser feita, é necessário calcular, inicialmente, a quantidade de meios de transportes ($valMT_{mt}$) que deve ser gerada na simulação, de acordo com os percentuais estabelecidos em $PCMT_{c,mt}$, para que os totais requeridos em $D_{d,c}$ sejam alcançados.

O algoritmo apresentado no Quadro 3.10 parte do pressuposto que o vetor $PMT_{c,mt}$ esteja classificado em ordem descendente por tamanho da capacidade do meio de transporte *mt*. Isto significa que os totais de meios de transportes, atribuídos em $valMT_{mt}$, são calculados primeiro para os meios de transporte com maior capacidade. Decorre desta premissa que a diferença entre $(\sum_{mt=1}^{QMT} valMT_{mt} * CMT_{mt})$ e $D_{d,c}$ será a menor possível. Visto que a compreensão do algoritmo abaixo é imediata, comenta-se apenas o artifício na linha 9 que, ao somar 0.5 ao total calculado de unidades para o último meio de transporte, tem por objetivo realizar um arredondamento para cima. Mesmo que a quantidade de unidades de meios de transporte calculada seja, por exemplo, igual a 2,1, a quantidade gerada no algoritmo será de 3.

Início

1 **Se** $D_{d,c} \neq 0$ **faça**

2 totalGerado = 0

3 **Para** mt = 1 **até** QMT **faça**

4 **Se** mt < QMT **faça**

5 $$valMT_{mt} = arredondar\left(\frac{D_{d,c} * PMT_{c,mt}}{CMT_{mt}}\right)$$

6 $$totalGerado = totalGerado + (valMT_{mt} * CMT_{mt})$$

7 **Senão**

8 $$valMT_{mt} = \left(\frac{D_{d,c} - totalGerado}{CMT_{mt}}\right)$$

9 $$valMT_{mt} = arredondar(valMT_{mt} + 0.5)$$

10 **Fim se**

11 **Fim para**

12 Converta $VT_{p,d,c}$ para $VTMT_{p,d,c,mt}$ de acordo com as quantidades definidas em $valMT_{mt}$

13 **Fim se**

Fim

Quadro 3.10 – Algoritmo de conversão de $VT_{p,d,c}$ para $VTMT_{p,d,c,mt}$

Feito o cálculo do vetor $valMT_{mt,}$ o próximo passo (linha 12) é o de calcular de que forma estas quantidades estarão distribuídas dentro dos períodos de planejamento, respeitando no máximo possível os valores definidos em $VT_{p,d,c}$. Supondo que a variável *QPP* indique a quantidade de períodos de planejamento programados, nos quais $VT_{i,d,c} \neq 0$ (para todo i =1...*QPP*) – onde *i* =1 é o primeiro período *p* com planejamento programado -, o problema de programação linear inteira (PPLI), apresentado no Quadro 3.11, realiza a conversão necessária.

O PPLI do Quadro 3.11 realiza a conversão de $VT_{p,d,c}$ para $VTMT_{p,d,c,mt}$ por meio de *goal programming*. O objetivo é fazer com que i) o somatório dos meios de transporte, distribuídos entre todos os períodos (*i*=1...*QPD*), seja igual à $valMT_{mt}$ - restrição (3.23)- e, ao mesmo tempo, ii) sejam minimizadas as diferenças, em excesso, entre o valor planejado $VT_{i,p,c}$ e o que realmente vai ocorrer em função da quantidade de meios de transportes gerados para aquele período *i*, em cada período do planejamento – restrição (3.22) em conjunto com a função objetivo (3.21). XP_i é a variável que mede esta diferença no período *i*.

Minimizar $Z = \sum_{i=1}^{QPP} XP_i$ (3.21)

Sujeito a:

$$\left(\sum_{i=1}^{QPP} \left(VTMT_{i,d,c,mt} * CMT_{mt} \right) \right) + XN_i - XP_i = VT_{i,d,c} \quad (3.22)$$

$$\sum_{i=1}^{QPP} VTMT_{i,d,c,mt} = valMT_{mt} \quad (3.23)$$

---------- variável inteira ----------

$VTMT_{i,d,c,mt}$, $\forall\ i = 1....QPP$

$\forall\ d = 1....PD$

$\forall\ c = 1....C$

$\forall\ mt = 1....QMT$

Quadro 3.11 – Algoritmo de cálculo de $VTMT_{p,d,c,mt}$ em função de $valMT_{mt}$

Considerando que $valMT_{mt}$={249,65}, CMT_{mt}={36,324}, PMT_{mt}={0.3, 0.7} e $VT_{i,d,c}$={3000, 3000, 3000, 3000, 3000, 3000, 3000, 3000, 3000, 3000}, dispõe-se, então, de dois meios de transporte com capacidades de 36 e 324 unidades e há um planejamento de 10 períodos consecutivos de 3000 unidades. Este planejamento será atendido por 249 e 65 unidades de meios de transporte mt=1 e mt=2, respectivamente. O Quadro 3.12 mostra os resultados do algoritmo de conversão dos valores $VT_{p,d,c}$ para $VTMT_{p,d,c,mt}$.

	Totais por período				
	Unidades de MTs		Unidades Demandadas		
Período	**mt=1**	**mt=2**	**mt=1**	**mt=2**	**Totais**
1	65	2	2340	648	2988
2	3	9	108	2916	3024
3	3	9	108	2916	3024
4	2	9	72	2916	2988
5	2	9	72	2916	2988
6	2	9	72	2916	2988
7	3	9	108	2916	3024
8	2	9	72	2916	2988
9	83		2988		2988
10	84		3024		3024
Totais	249	65	8964	21060	30024

Quadro 3.12 – Resultados do algoritmo de conversão de $VT_{p,d,c}$ para $VTMT_{p,d,c,mt}$

Observe-se que as demandas de 3000 unidades por período foram respeitadas, no máximo possível, visto que as capacidades dos meios de transportes não são múltiplos das demandas. Além, disso o total geral planejado, 30024 unidades, é a melhor solução que atende à demanda. Se for retirada uma unidade do meio de transporte de menor capacidade (36), a demanda necessária não será atingida. Além disso, os percentuais para cada meio de transporte foram respeitados no máximo possível.

3.3.3.3 Gerando os eventos em função das taxas $VTMT_{p,d,c,mt}$

Diferentemente dos métodos propostos nos itens 3.1 e 3.2, onde as taxas definidas pelo projetista do modelo (vetor *R*) estipulam valores médios para cada intervalo *k*, os valores de $VTMT_{p,d,c,mt}$ estabelecem a quantidade exata de meios de transporte que devem ser gerados de modo que a demanda no período *d* seja atendida. Desta forma, o algoritmo de geração dos eventos, apesar de garantir o ajuste à distribuição (dado por Φ) no que concerne aos intervalos de chegada entre eventos, garante que o total de eventos gerados será igual à $VTMT_{p,d,c,mt}$. Consequência deste aspecto é que o tamanho do vetor $T_{d,c,mt}$ assume, inicialmente, o valor de $\sum_{p=1}^{P} VTMT_{p,d,c,mt}$ (onde $P = PA + PD$) e não mais precisa ser redimensionado.

O Quadro 3.13 apresenta o algoritmo de geração de eventos em função das taxas $VTMT_{p,d,c,mt}$. Da mesma forma que nos casos anteriores, entre as linhas 7 e 14, são gerados aleatoriamente $VTMT_{k,\ d,c,mt}$ eventos, distribuídos de acordo com a função Φ, após o instante B_{k-1}. Já que a geração dos intervalos de chegada é aleatória, é possível que o limite superior B_k seja ultrapassado. Caso isto ocorra, os $VTMT_{k,\ d,c,mt}$ eventos gerados são ajustados (linhas 18 a 22) a fim de que caibam dentro do intervalo $[B_{k-1}, B_k]$, mantendo-se a distribuição entre os intervalos de chegada produzida por Φ.

Início DirectAlgorithmTaxasCalculadas($T_{d,c,mt}$, d, c, mt))

1 Dimensione o vetor $T_{d,c,mt}$ com o tamanho $\sum_{p=1}^{P} VTMT_{p,d,c,mt}$

2 n = 0

3 **Para** k = 1 **até** P **faça**

4 **Se** $VTMT_{k,d,c,mt} \neq 0$ **faça**

5 tnAnt = B_{k-1}

6 iP = n + 1

7 **Para** i = 1 **até** $VTMT_{k,d,c,mt}$ **faça**

8 n = n + 1

9 Gere um número aleatório $u \sim U[0,1]$

10 A = $\Phi(u)$

11 W = A / $VTMT_{k,d,c,mt}$ * ($B_k - B_{k-1}$)

12 $T_{d,c,mt,n}$ = tnAnt + W

13 tnAnt = $T_{d,c,mt,n}$

14 **Fim para**

15 **Se** $T_{d,c,mt,n} > B_k$ **faça**

16 ajusteAnt = B_{k-1}

17 naoAjusteAnt = B_{k-1}

18 **Para** i = iP **até** n **faça**

19 $temp = ajusteAnt + \frac{(T_{d,c,mt,i} - naoAjusteAnt)}{T_{d,c,mt,n} - B_{k-1}} * (B_k - B_{k-1})$

20 $naoAjusteAnt = T_{d,c,mt,i}$

21 $T_{d,c,mt,i} = temp$

22 $ajusteAnt = T_{d,c,mt,i}$

22 **Fim para**

23 **Fim se**

24 **Se** n = $C_{d,c,mt,P}$ **faça**

25 Sair laço **Para**

26 **Fim se**

27 **Fim se**

28 **Fim para**

29 Retorne $T_{d,c,mt}$

Fim

Quadro 3.13 - Algoritmo *DirectAlgorithmTaxasCalculadas* para PR com taxas calculadas

3.3.4 Resultados do Algoritmo Proposto

A implementação do algoritmo '*Algoritmo Principal*', apresentado no Quadro 3.9, foi feita no *Microsoft Visual Basic 6.3*, disponível no *Arena*. A implementação dos passos 3 e 4 desse algoritmo foi feita usando o *Lingo 6.0*.

Considerem-se os seguintes dados de entrada do Quadro 3.14 onde os valores de *CE*, *CT*, *LSMin*, *LSMax*, *CMT* e *D* são fornecidos em toneladas. Neste exemplo, o planejamento da programação deve ser feito por meio de 2 meios de transporte (*caminhão* e *trem*) com capacidades de 36 e 324 toneladas, respectivamente. Há 30 períodos de demanda de 3 classes distintas de produtos. A programação deve ser feita de modo que as mercadorias estejam armazenadas com 2 períodos de antecedência (PS=2). A capacidade de recebimento por período é de 13000 toneladas e a capacidade estática é de 96000 toneladas.

- **Pesos** (*p1* = 1, *p2* = 2, *p3* = 3, *p4* = 4)
- ***CE***: 96000;
- ***CT***: 13000;
- ***PS***: 2;
- ***PD***: 30;
- ***PA***: 14;
- ***LSMin***: 300;
- ***LSMax***: 3000;
- **C**: 3;
- NC_c: {classe1, classe2, classe3}
- ***QMT***: 2;
- NMT_{mt}: {caminhão, trem};
- CMT_{mt}: {36, 324};
- $PMT_{c,mt}$: $\begin{bmatrix} 0.3 & 0.7 \\ 0.5 & 0.5 \\ 0.7 & 0.3 \end{bmatrix}$
- **Φ**: -ln(u);

$D_{d,c}$:

	Classes 1 Classe1	2 Classe2	3 Classe3
1	10,000	10,000	10,000
2	5,000	-	7,000
3	10,000	10,000	10,000
4	-	-	-
5	10,000	20,000	10,000
6	-	-	-
7	5,000	10,000	20,000
8	-	-	-
9	2,000	5,000	3,000
10	10,000	5,000	25,000
11	10,000	10,000	10,000
12	-	-	-
13	2,000	2,000	6,000
14	-	-	-
15	12,000	8,000	20,000
16	-	-	-
17	-	-	-
18	20,000	-	-
19	12,000	1,000	3,000
20	-	-	-
21	-	-	-
22	-	40,000	-
23	-	-	-
24	20,000	-	20,000
25	30,000	-	-
26	-	-	-
27	-	-	-
28	5,000	-	-
29	-	-	-
30	-	-	-
Total	163,000	121,000	144,000

Quadro 3.14 – Dados de entrada

Conforme exposto anteriormente no item 3.3.3, o algoritmo proposto gera o vetor *T* cujas entradas T_i tem o formato: tempo de chegada, *mt*, *c*, *d*. O Quadro 3.15 resume os totais de eventos gerados.

O tempo computacional do *'Algoritmo Principal'* foi de 2min07s , com os seguintes tempos parciais:

- Passo 3 (planejamento da programação) → 51s;
- Passo 4 (conversão dos valores $VT_{p,d,c}$ para $VTMT_{p,d,c,mt}$) → 1min15s;

- Passo 5 ao 11 (geração dos tempos de chegada) → < 1s;

- Passos 12 e 13 (geração do vetor T) → < 1s.

Período Demanda	classe 1					classe 2				
	Qtde (unidades)		Qtde (Ton)		Total Ton	Qtde (unidades)		Qtde (Ton)		Total Ton
	caminhão	Trem	Caminhão	Trem		caminhão	Trem	Caminhão	Trem	
1	80	22	2880	7128	10008	143	15	5148	4860	10008
2	40	11	1440	3564	5004					
3	80	22	2880	7128	10008	143	15	5148	4860	10008
5	80	22	2880	7128	10008	277	31	9972	10044	20016
7	40	11	1440	3564	5004	143	15	5148	4860	10008
9	20	4	720	1296	2016	67	8	2412	2592	5004
10	80	22	2880	7128	10008	67	8	2412	2592	5004
11	80	22	2880	7128	10008	143	15	5148	4860	10008
13	20	4	720	1296	2016	29	3	1044	972	2016
15	100	26	3600	8424	12024	115	12	4140	3888	8028
18	169	43	6084	13932	20016					
22						554	62	19944	20088	40032
24	169	43	6084	13932	20016					
19	100	26	3600	8424	12024	10	2	360	648	1008
25	249	65	8964	21060	30024					
28	40	11	1440	3564	5004					
(vazio)										
Total geral	1347	354	48492	114696	163188	1691	186	60876	60264	121140

Período Demanda	classe3				
	Qtde (unidades)		Qtde (Ton)		Total Ton
	caminhão	Trem	Caminhão	Trem	
1	197	9	7092	2916	10008
2	141	6	5076	1944	7020
3	197	9	7092	2916	10008
5	197	9	7092	2916	10008
7	385	19	13860	6156	20016
9	57	3	2052	972	3024
10	488	23	17568	7452	25020
11	197	9	7092	2916	10008
13	113	6	4068	1944	6012
15	385	19	13860	6156	20016
18					
22					
24	385	19	13860	6156	20016
19	57	3	2052	972	3024
25					
28					
(vazio)					
Total geral	2799	134	100764	43416	144180

Quadro 3.15 – Totais de eventos gerados em função das demandas necessárias

Nota-se que os totais obtidos para cada demanda não são exatos, visto que as unidades de demanda e as capacidades dos meios de transporte não são múltiplos. No entanto,

os valores obtidos são o mais próximo possível de cada demanda. Caso seja retirada apenas uma unidade do meio de transporte de menor capacidade, isto já seria suficiente para que a quantidade requerida naquele período não seja atendida.

3.3.5 Algoritmo de Geração dos Eventos usando a função Beta Generalizada

Considerando os problemas apontados no item 2.3.3.3 do referencial teórico, bem como a possibilidade de não ser possível determinar o formato da distribuição dos intervalos de chegada, este item apresenta uma variação do algoritmo *DirectAlgorithmTaxasCalculadas*, usando a função *Beta Generalizada.*

Normalmente, a função *Beta* é usada quando se está modelando um problema na ausência de dados. Sendo uma variável aleatória *X* distribuída, em um intervalo [*a, b*], de acordo com uma distribuição *Beta Generalizada*, o Quadro 3.16 apresenta as informações relevantes dessa distribuição, onde Γ representa a função *Gama,* $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$.

$$f_X(x;a,b,\alpha_1,\alpha_2) = \frac{\Gamma(\alpha_1+\alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)(b-a)^{\alpha_1+\alpha_2-1}}(x-a)^{\alpha_1-1}(b-x)^{\alpha_2-1} \quad (3.24)$$

$$E[X] = \frac{\alpha_1 b + \alpha_2 a}{\alpha_1 + \alpha_2} \quad (3.25)$$

$$VAR[X] = \frac{(b-a)^2 \alpha_1 \alpha_2}{(\alpha_1+\alpha_2)^2(\alpha_1+\alpha_2+1)} \quad (3.26)$$

$$Moda = \frac{(\alpha_1-1)b + (\alpha_2-1)a}{\alpha_1+\alpha_2-2}, \text{ se } \alpha_1>1 \text{ e } \alpha 2>1 \quad (3.27)$$

Quadro 3.16 – Informações da Distribuição Beta

A função *Beta Generalizada* é uma função de parâmetros *a*, *b*, α_1 e α_2 e apresenta diferentes formatos, conforme ilustra a Figura 3.12. É importante notar que, para uma *fdp* $f_X(.)$ contínua, a moda é um máximo local da função e se existe um único máximo global para $f_X(.)$, então a *fdp* é dita ser unimodal. Para a função *Beta*, se $\alpha_1>1$ e $\alpha 2>1$, ela é unimodal.

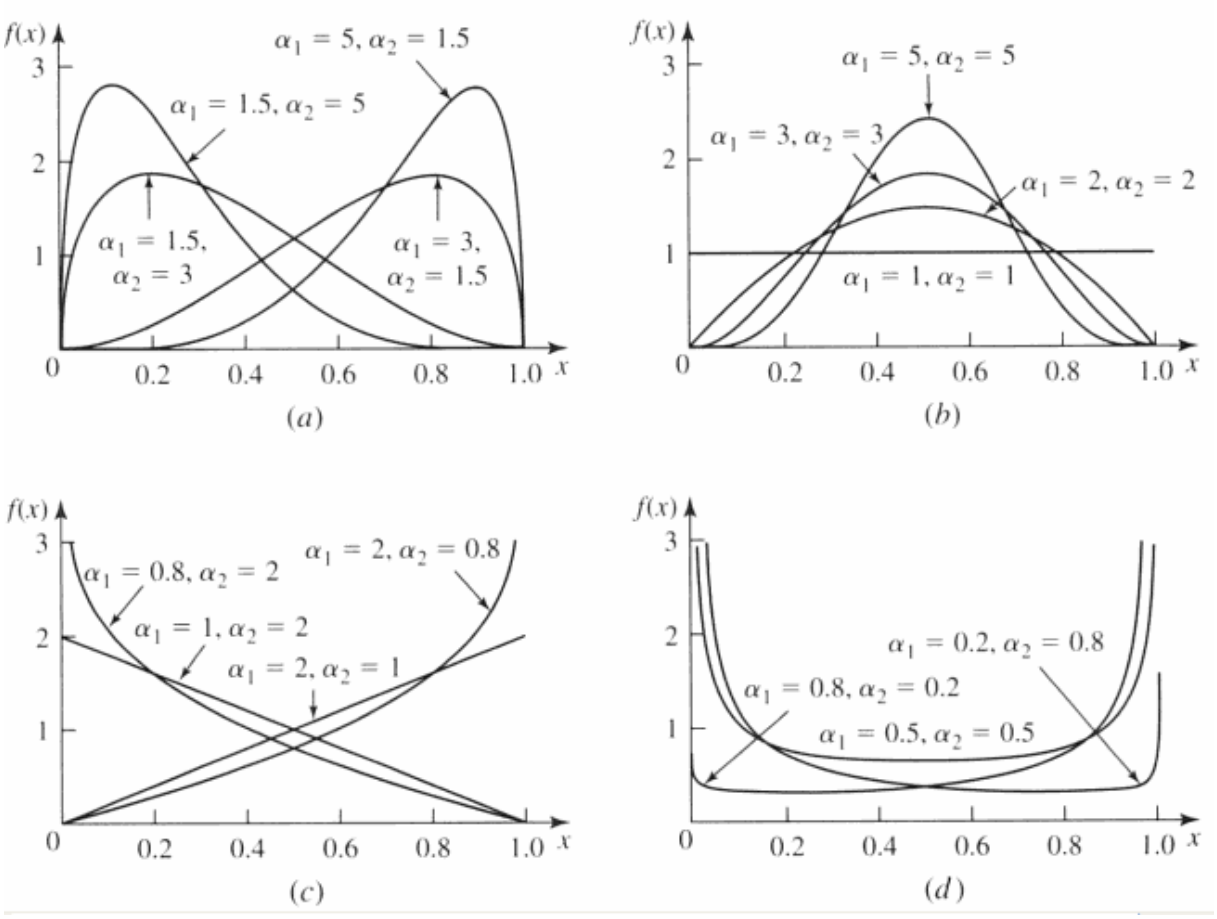

Figura 3.12 – Distribuição *Beta* de acordo com parâmetros $a = 0$, $b = 1$

Fonte: (LAW & KELTON; 2000, p. 309)

O Quadro 3.17 mostra o algoritmo de geração de eventos distribuídos de acordo com a função *Beta Generalizada*, usando como parâmetros de entrada os valores $a = 0$, $b = 24$, $\alpha_1 = 2$ e $\alpha_2 = 8$, *txBeta* = 500, cujo gráfico desta função com estes parâmetros é exibido na Figura 3.13.

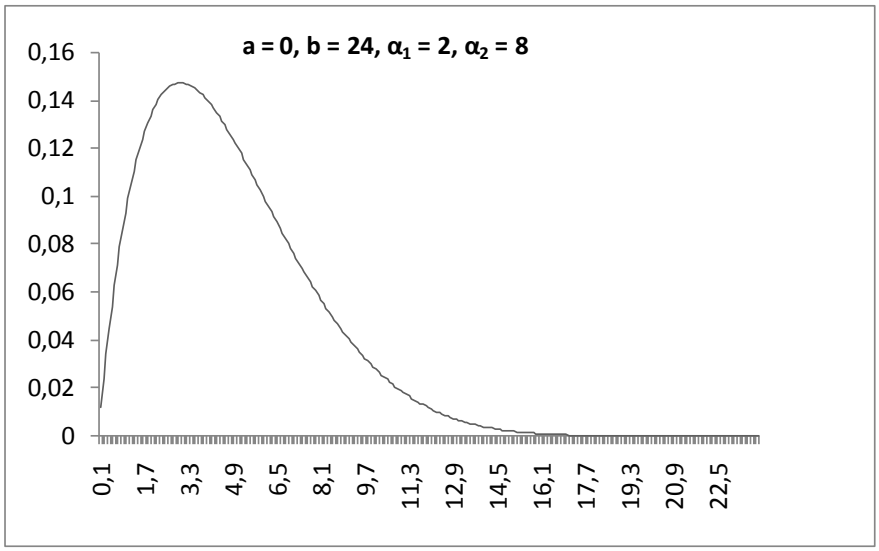

Figura 3.13 – Distribuição Beta com parâmetros a = 0, b = 24, α_1 = 2, α_2 = 8

geraEventosBetaGeneralizadaComTaxa(a, b, α_1, α_2, txBeta)

1 moda = $\dfrac{(\alpha_1 - 1)b + (\alpha_2 - 1)a}{\alpha_1 + \alpha_2 - 2}$
2 limSup = f_x(moda, a, b, α_1, α_2)
3 txPeriodo = txBeta * limSup
4 n = 0, Tn = 0, k = 0
5 **Faça**
6 n = n + 1, u ~U[0,1]
7 Z = -ln(u)
8 Z = Z / txPeriodo
9 Tn = Tn + Z, u ~U[0,1]
10 U = u * limSup
11 **Se** (U <= (f_x(T_n, a, b, α_1, α_2))) **Então**
12 k = k + 1
13 Escolher Tn
14 **Fim Faça**
15 **Fim Faça Enquanto** Tn < b
Fim

Quadro 3.17 – Algoritmo de geração de eventos usando a função *Beta Generalizada* por meio de taxa *txBeta*

O algoritmo inicia calculando a *moda* que para esses parâmetros é igual a 3 e o ponto máximo da função *Beta, limSup* = 0.147. Então, a *txPeriodo* será igual a 73.63. Isso significa que, em média, serão gerados no total 73.63 * 24 = 1767.13 eventos no período compreendido entre *a* e *b*. Entre as linhas 5 e 15, o algoritmo funciona de acordo com o método da

aceitação/rejeição. É gerado um evento *U* entre 0 e *limSup* e o evento *Tn* somente será selecionado se o valor de *U* estiver abaixo da $f_x(T_n, a, b, \alpha_1, \alpha_2)$. Considerando que a área total acima da reta definada pelo limite superior da função *Beta* (*limSup* * (*b* – *a*)) é igual a 3.53 e que a área abaixo da $f_x(T_n, a, b, \alpha_1, \alpha_2)$ é igual a unidade, tem-se que, em média, serão selecionados, 28 % de eventos e esses estarão distribuídos de acordo com f_x.

Executou-se o algoritmo para 1000 replicações e obteve-se para *n* o valor médio de 1768.31 com IC = 2.70 a 95% e para *k*, o valor médio de 500.387 com IC = 1.43.

Entretanto, como as taxas $VTMT_{k,d,c,mt}$ não são valores médios, mas sim a exata quantidade de eventos que deve ser gerada, o algoritmo acima foi modificado para que se obtenha uma quantidade determinada de eventos.

O Quadro 3.18 exibe o algoritmo modificado para produzir a exata quantidade de $VTMT_{k,d,c,mt}$ eventos. Para cada iteração, geram-se dois eventos, um no eixo *x* (T_n~*U* [a, b]) e outro no eixo *y* (U~[0, *limSup*]). Da mesma forma, o evento é selecionado caso o valor *U* seja menor ou igual ao valor de $f_x(T_n, a, b, \alpha_1, \alpha_2)$. Executaram-se 1000 replicações do algoritmo (com os mesmos dados de entrada), e obteve-se para *n* o valor médio de 1765.83 com IC = 4.04 a 95% e para *k*, como era o objetivo, obteve-se sempre 500 eventos. Ou seja, a eficiência de ambos os procedimentos é a mesma. Para se obter o mesmo número de eventos em ambos os algoritmos, são necessários, em média, o mesmo número de iterações. A diferença é que os valores selecionados (*Tn*) não mais estarão ordenados. Recomenda-se, então, o uso de uma estrutura de *heap*[8] (HOROWITZ; SAHNI; ANDERSON-FREED, 2007) para o armazenamento dos valores selecionados.

[8] Um *heap* é uma estrutura de dados em árvore binária que satisfaz a condição de que um nodo filho sempre armazena valores menores ou iguais ao nodo pai. Portanto, a raiz da árvore tem sempre o valor mais alto. A condição pode ser invertida para que no nodo raiz sempre tenha o menor valor, permitindo uma retirada dos nodos da árvore em ordem ascendente. A estrutura *heap* compete com algoritmos de classificação por apresentar ordem de complexidade O(n log n) para o pior caso, sendo bastante atrativa quando o tamanho dos dados é muito grande. Lembrar que o *QuickSort* tem ordem quadrática de complexidade $O(n^2)$ para o pior caso.

geraQtdeFixaDeEventosBetaGeneralizada (a, b, α_1, α_2, $VTMT_{k,d,c,mt}$)

1 moda = $\frac{(\alpha_1 - 1)b + (\alpha_2 - 1)a}{\alpha_1 + \alpha_2 - 2}$
2 limSup = f_x(moda, a, b, α_1, α_2)
3 n = 0, Tn = 0, k = 0
4 **Faça**
5 n = n + 1,
6 u ~U[0,1]
7 Tn = a + u * (b – a)
8 u ~U[0,1]
9 U = u * limSup
10 **Se** (U <= (f_x(T_n, a, b, α_1, α_2))) **Então**
11 k = k + 1
12 Escolher Tn
13 **Fim Faça**
14 **Fim Faça Enquanto** k < $VTMT_{k,d,c,mt}$
Fim

Quadro 3.18 – Algoritmo de geração de $VTMT_{k,d,c,mt}$ eventos usando a função *Beta Generalizada*

Kuhl et al. (2008, p. 65) mostra os estimadores para distribuição *Beta Generalizada*, obtidos por meio do método dos momentos, considerando uma amostra $\{X_i: i = 1, ..., n\}$ de tamanho *n*. Sendo $X_{(1)} \leq X_{(2)} \leq ... \leq X_{(n)}$ a amostra ordenada dos valores de X_i, a seguir são exibidos os estimadores.

$$\hat{a} = 2*X_{(1)} - X_{(2)} \tag{3.28}$$

$$\hat{b} = 2*X_{(n)} - X_{(n-1)} \tag{3.29}$$

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \,, \tag{3.30}$$

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2 \tag{3.31}$$

Os estimadores $\hat{\alpha}_1$ e $\hat{\alpha}_2$ são obtidos por meio dos valores d_1 e d_2 calculados conforme a seguir.

$$d_1 = \frac{\left(\overline{X} - \hat{a}\right)}{\left(\hat{b} - \hat{a}\right)} \tag{3.32}$$

$$d_2 = \frac{S}{\left(\hat{b} - \hat{a}\right)} \tag{3.33}$$

$$\hat{\alpha}_1 = \frac{d_1^2\left(1 - d_1\right)}{d_2^2} - d_1 \tag{3.34}$$

$$\hat{\alpha}_2 = \frac{d_1\left(1 - d_1\right)^2}{d_2^2} - \left(1 - d_1\right) \tag{3.35}$$

Executou-se uma simulação com 1000 replicações, usando os parâmetros a = 0, b = 24, α_1 = 2, α_2 = 8, e $VTMT_{k,d,c,mt}$ = 5000. Para cada replicação, calculou-se a média e a variância dos valores X_i ´s e os estimadores $\hat{\alpha}_1$ e $\hat{\alpha}_2$, de acordo com (3.34) e (3.35). O Quadro 3.19 compara os valores esperados para a distribuição *Beta Generalizada* com os valores médios obtidos das 1000 replicações, comprovando que o método de geração convergiu para os valores esperados.

	E[X]	S^2	α_1	α_2
Valores Esperados	4.8	8.378	2	8
Valores médios Calculados	4.8005	8.3775	2.00135	8.005
IC(95%)	(4.798, 4.803)	(8.366, 8.389)	(1.999, 2.004)	(7.994, 8.016)

Quadro 3.19 – Comparação com os valores esperados com os valores médios de 1000 replicações

3.3.6 Metodologia de Aplicação do Método

A Figura 3.14 apresenta a metodologia de aplicação do método proposto. A finalidade principal é mostrar que a implementação do '*Algoritmo Principal*' (Quadro 3.9) não precisa ser feita de forma monolítica. Dependendo dos objetivos do projetista e, principalmente, do tamanho do problema (quantidade de destinos e tamanho das variáveis *PD, PA, C, QMT*), a implementação pode ser feita de forma separada da simulação. Inclusive, a fase 4 (Gerar sequência aleatória) pode ser executada anteriormente à simulação, fornecendo-se diretamente a sequência aleatória necessária para a simulação.

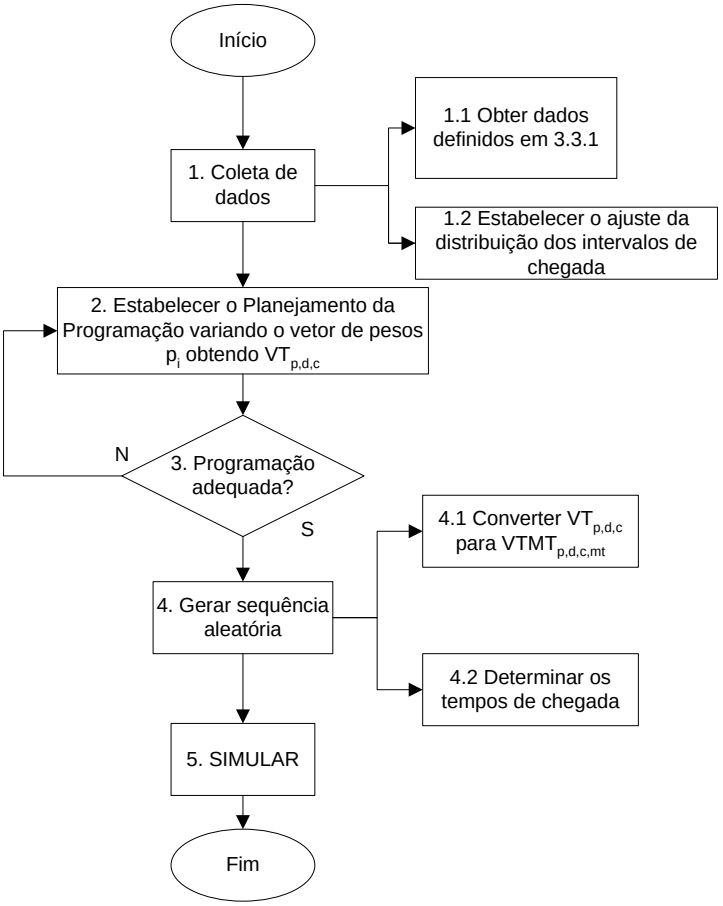

Figura 3.14 – Metodologia de Aplicação do Método

4 APLICAÇÃO DOS MÉTODOS PROPOSTOS PARA GERAÇÃO DE ENTIDADES EM AMBIENTES DE SIMULAÇÃO

Este capítulo visa mostrar a aplicação dos métodos propostos nas áreas da Administração (*call center* e logística) e Ciência da Computação (redes de computadores).

4.1 APLICAÇÃO DOS MÉTODOS NA ADMINISTRAÇÃO – *CALL CENTER*

As centrais de atendimento ao público (*call centers*) têm despertado o interesse dos pesquisadores em virtude do crescimento de sua adoção por parte tanto de empresas privadas quanto de públicas. Barboza *et al.* (2003) aplicam as técnicas de simulação, programação linear e o algoritmo de *matching* a fim de determinar a quantidade necessária de atendentes, por período, considerando a demanda esperada de chamados e a preferência dos funcionários por determinados horários. No que tange à simulação, o trabalho foi desenvolvido usando linguagem de programação de uso geral (*Visual Basic*) e, como já foi dito, coube aos autores implementar todas as funções necessárias para a realização da simulação, inclusive a geração de números aleatórios.

Uma central de atendimento ao público em uma organização é um sistema que disponibiliza um número telefônico central por meio do qual os clientes podem fazer chamadas para suporte técnico, compras, acompanhamento das ordens de compras, dúvidas, reclamações, sugestões, etc. A este número de telefone central está associada uma determinada quantidade de linhas telefônicas. Quando houver uma chamada e todas estas linhas já estiverem em uso, o cliente ou usuário receberá um sinal de ocupado e deverá repetir a ligação mais tarde. Caso tenha obtido sucesso na chamada, mensagens eletrônicas descrevem as opções disponíveis para o cliente que, por meio do teclado numérico do telefone, faz suas escolhas. Atualmente, a nova geração de centrais de atendimento possui o recurso de reconhecimento de voz, o que dispensa o teclado numérico.

Os administradores de uma central de atendimento devem estabelecer, criteriosamente, qual é a quantidade de linhas telefônicas que serão contratadas e quantos funcionários, em cada período do dia, deverão ser alocados para cada tipo de atendimento (vendas, suporte técnico, reclamações, problemas com faturas, etc.). Devido à complexidade organizacional, seria impossível, exceto com um custo elevadíssimo, treinar todos os atendentes para todos os tipos de chamados. É possível, portanto, que o cliente, apesar de ter conseguido fazer a ligação (havia pelo menos uma linha disponível na central), não consiga ser atendido naquele exato momento, pois todos os atendentes que tratam do seu assunto (suporte técnico, por exemplo) já estão ocupados com outras chamadas. Se houver uma demora muito grande para que o atendimento ocorra, o cliente desistirá da chamada, aumentando, com certeza, sua insatisfação com o sistema. Tanto este tipo de desistência quanto a quantidade de chamadas que obtiveram linhas ocupadas devem ser atentamente monitoradas para que se possa determinar a quantidade exata de linhas e de atendentes por período para cada tipo de chamada.

Kelton, Sadowski e Sadowski (2001) apresentam um modelo de simulação, implementado no *Arena Simulation*, para uma central de atendimento ao público. Como neste sistema, as chegadas de entidades (chamadas) se dão por um processo não-estacionário (as taxas de chegada variam por período), toda a complexidade da geração de clientes é tratada dentro do próprio modelo pelos projetistas.

Foge do escopo desta pesquisa discutir internamente o modelo em si e seus resultados, já amplamente apresentados no capítulo 5 em [KELTON, SADOWSKI e SADOWSKI, 2001]. O interesse neste ponto é nas diferenças entre a modelagem original utilizada pelos autores e na utilização dos métodos propostos para geração de entidades.

4.1.1 Modelo de Simulação Original de *Call Center*

O modelo de *call center* é implementado no *Arena* usando o recurso *submodel* (submodelo), conforme mostra a Figura 4.1. O submodelo "*Create and Direct Arrivals*" implementa a lógica de controle das chegadas das chamadas que são, em seguida, distribuídas para os próximos submodelos de acordo com o tipo de chamada (*Technical Call, Sales Call* e *Order-Status Call*).

Figura 4.1- Lógica do modelo original usando *submodel*

O submodelo "*Create and Direct Arrivals*" (Figura 4.2) implementa a lógica de controle das chegadas das chamadas.

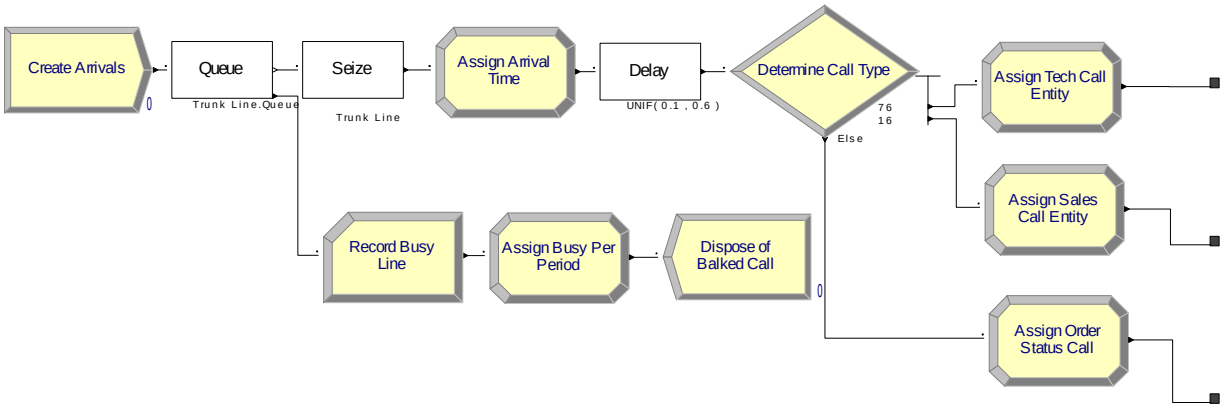

Figura 4.2 - Submodelo "*Create and Direct Arrivals*"

O módulo "*Create Arrivals"* gera as entidades ("*Arrival Entity*") de acordo com o "*Arrival Schedule"* definido conforme as taxas do Quadro 4.1. O *Arena* gera as entidades dentro de cada intervalo assumindo que as chegadas são *Poisson*.

Período	*Tempo*	*Taxa*	*Período*	*Tempo*	*Taxa*
1	08:00 - 08:30	20	11	13:00 - 13:30	110
2	08:30 - 09:00	35	12	13:30 - 14:00	95
3	09:00 - 09:30	45	13	14:00 - 14:30	105
4	09:30 - 10:00	50	14	14:30 - 15:00	90
5	10:00 - 10:30	70	15	15:00 - 15:30	85
6	10:30 - 11:00	75	16	15:30 - 16:00	90
7	11:00 - 11:30	75	17	16:00 - 16:30	70
8	11:30 - 12:00	90	18	16:30 - 17:00	65
9	12:00 - 12:30	95	19	17:00 - 17:30	45
10	12:30 - 13:00	105	20	17:30 - 18:00	30

Quadro 4.1 – Taxas de chegada das chamadas (por hora)

Caso não haja uma linha disponível para a chamada, o módulo "*Queue"* encaminha-a para o registro de uma chamada não atendida ("*Balked Call*"). O módulo "*Assign Arrival Time*" registra o tempo de chegada da chamada e o módulo "*Delay"* aguarda um tempo u~U[0.1, 0.6], representando o tempo que o cliente leva para optar pelo seu tipo de chamada. O módulo "*Determine Call Type*" distribui as chamadas, para os submodelos correspondentes, na proporção de 76% para "*Technical Call*", 16% para "*Sales Call*" e 8% para "*Order Status Call*". Só a partir deste instante é que a entidade passa a ser "vista" dentro do modelo como uma chamada específica – até este momento era uma entidade do tipo "*Arrival Entity*". Consequentemente, o registro estatístico das chamadas não atendidas não pode ser feito por tipo de chamada, mas apenas como uma "*Arrival Entity*".

4.1.2 Modelo de simulação modificado de *Call Center*

A Figura 4.3 mostra o modelo modificado usando o template para processos de renovação com taxas constantes por período, de uma origem para múltiplos destinos ("PR1N"). Observe-se que foi mantido o modelo original (em inglês) adaptando-o no que necessário.

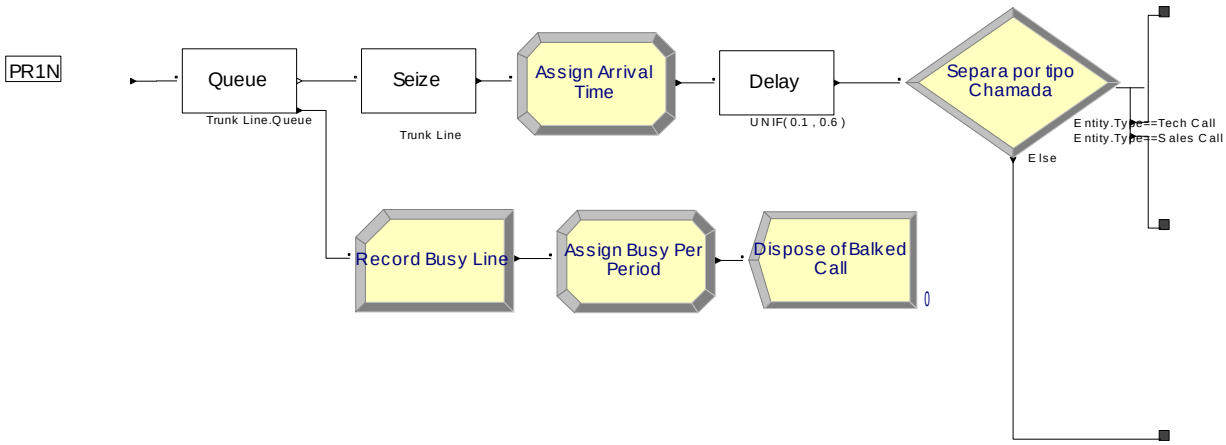

Figura 4.3 - Submodelo "*Create and Direct Arrivals*" modificado

Levando em consideração o modelo original, foram usados os seguintes dados de entrada:

- $\boldsymbol{P}$ = 20;
- $\boldsymbol{C}$ = 3;
- $\boldsymbol{NC_c}$ = { '*Technical Call*', '*Sales Call'*, '*Order Status Call'*}
- $R_{c,p}$ =

Classes	Períodos																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Technical Call	7,6	13,3	17,1	19,0	26,6	28,5	28,5	34,2	36,1	39,9	41,8	36,1	39,9	34,2	32,3	34,2	26,6	24,7	17,1	11,4
Sales Call	1,6	2,8	3,6	4,0	5,6	6,0	6,0	7,2	7,6	8,4	8,8	7,6	8,4	7,2	6,8	7,2	5,6	5,2	3,6	2,4
Order-Status Call	0,8	1,4	1,8	2,0	2,8	3,0	3,0	3,6	3,8	4,2	4,4	3,8	4,2	3,6	3,4	3,6	2,8	2,6	1,8	1,2

- $\boldsymbol{B_p}$ = {0, 30, 60, 90, 120, 150, 180, 210, 240, 270, 300, 330, 360, 390, 420, 450, 480, 510, 540, 570, 600} (em minutos);
- $\boldsymbol{\Phi} = -ln(u)$;

Desta forma, tem-se 20 períodos com 3 classes. Os valores de $R_{c,p}$ já são fornecidos considerando o percentual de cada classe (76% para "*Technical Call*", 16% para "*Sales Call*" e 8% para "*Order Status Call*"). Lembre que os valores do Quadro 4.1 são fornecidos (por hora) e que os períodos são de 30 minutos. A função $\boldsymbol{\Phi}$ é igual a $-ln(u)$ visto que o processo de renovação do *call center* é um processo *Poisson.*

4.1.3 Comparação entre os resultados

Para fins de comparação e validação do método proposto, foram executadas 1000 rodadas de simulação para ambos os modelos (original e modificado). O Quadro 4.2 exibe os resultados para os três primeiros períodos da simulação (08h00 às 09h30min) com o HW, valores máximos e mínimos observados entre as 1000 rodadas. Como se pode comprovar estatisticamente pelos resultados, a taxa de chegada para cada tipo de chamada converge para os respectivos valores esperados para ambos os modelos.

Entretanto, ressalte-se uma diferença importante entre os modelos da Figura 4.2 e da Figura 4.3. No modelo modificado, as entidades já são geradas de acordo com as taxas pré-

determinadas. No modelo original, o projetista precisa fazer a separação de acordo com os percentuais estabelecidos. Além de o modelo modificado ter se tornado mais simples, é preciso destacar que este está em maior sintonia com a realidade, pois quando ocorre uma chamada, já se sabe qual é o tipo de chamada – o cliente não vai tomar esta decisão somente após ter telefonado para o *call center*. Além disso, com o modelo modificado seria possível ter as estatísticas, também, das chamadas não atendidas por tipo de chamada.

			Modelo Original				Modelo Modificado			
C	*P*	*Valor Esperado*	*Média*	*HW*	*Mín*	*Max*	*Média*	*HW*	*Mín*	*Max*
1	1	7,60	7,61	0,17	1	21	7,58	0,17	0	17
	2	13,30	13,23	0,22	3	28	13,33	0,23	4	29
	3	17,10	16,80	0,24	5	29	17,23	0,26	4	32
	4	19,00	18,37	0,26	6	32	18,90	0,28	8	35
	5	26,60	25,67	0,28	13	43	26,71	0,31	13	45
2	1	1,60	1,66	0,08	0	7	1,65	0,08	0	7
	2	2,80	2,82	0,10	0	8	2,82	0,10	0	10
	3	3,60	3,55	0,11	0	10	3,57	0,12	0	10
	4	4,00	3,87	0,12	0	11	4,12	0,13	0	13
	5	5,60	5,39	0,14	0	13	5,71	0,15	0	14
3	1	0,80	0,77	0,06	0	5	0,83	0,06	0	5
	2	1,40	1,41	0,07	0	6	1,41	0,08	0	8
	3	1,80	1,80	0,08	0	7	1,92	0,08	0	6
	4	2,00	1,96	0,09	0	8	2,00	0,09	0	7
	5	2,80	2,77	0,10	0	9	2,77	0,10	0	8

Quadro 4.2 – Resultados do modelo original e modificado para os cinco primeiros períodos[9]

Caso os autores (KELTON; SADOWSKI; SADOWSKI, 2001) tivessem optado por uma implementação semelhante à proposta, eles teriam que criar, para cada tipo de chamada, um *Schedule* e a inserção correspondente de um módulo *Create*, o que, conforme já discutido no capítulo 2, é uma solução que não apresenta escalabilidade.

[9] Tempo Computacional
Modelo Original → 0,525s por rodada;
Modelo Modificado → 0,604s por rodada.

4.2 APLICAÇÃO DOS MÉTODOS EM TECNOLOGIA DE INFORMAÇÃO

Redes de computadores, de acordo com sua extensão geográfica, podem ser classificadas como *Local Area Networks* (LAN), *Metropolitan Area Networks* (MAN) e *Wide Area Networks* (WAN) (TANENBAUM, 2002).

Uma rede de longa distância (WAN), normalmente, está distribuída em uma área geográfica bastante dispersa (país ou continente) e é composta por um número muito grande de computadores (usualmente chamados de *hosts*), interligados por uma sub-rede de comunicação. A função da sub-rede é transmitir as mensagens que são enviadas entre os computadores ou *hosts*. Na maioria das redes WANs, a sub-rede é composta de dois elementos principais: linhas de transmissão e elementos de chaveamento, também conhecidos por roteadores (*routers*). Neste tipo de rede, conforme mostra a Figura 4.4, cada *host* é conectado diretamente a uma LAN, que por sua vez, está conectada à WAN por meio de um roteador.

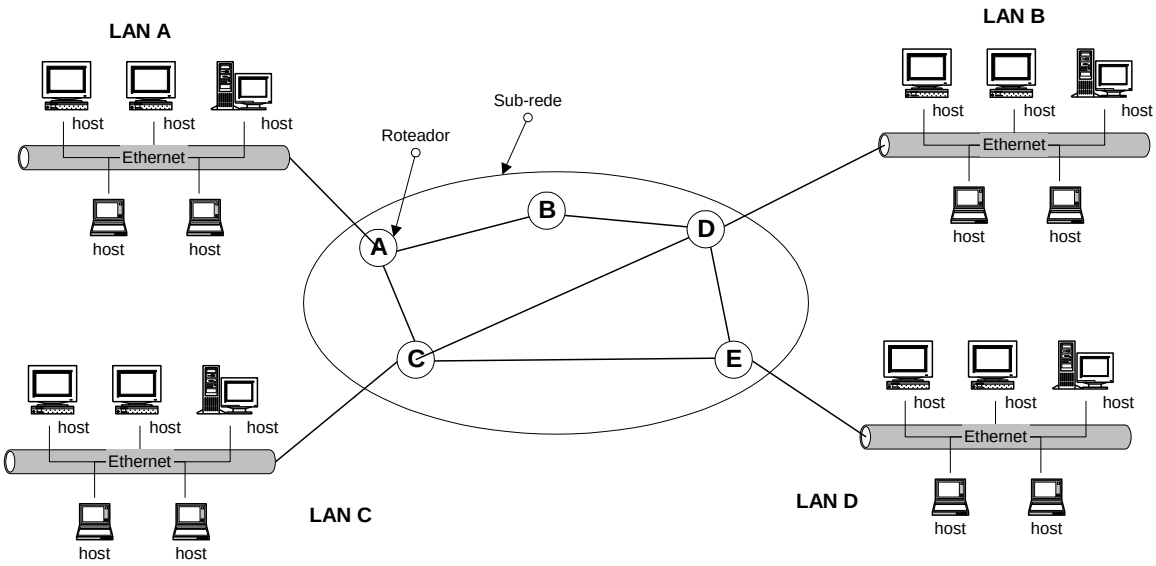

Figura 4.4 - Componentes de uma rede local de longa distância (WAN).

Uma WAN, em muitos casos, possui um grande número de linhas de transmissão, as quais individualmente conectam um par de roteadores. Quando dois roteadores não estão diretamente interligados e uma mensagem (pacote) necessita ser transferida de um para outro, a comunicação é feita indiretamente por outros roteadores.

Supondo que, na Figura 4.4, uma mensagem necessite ser transferida de um host localizado na LAN D para outro na LAN C, os seguintes caminhos dentro da sub-rede podem ser utilizados: DEC, DC e DBAC. Quando uma mensagem é enviada de um roteador para outro, por meio de roteadores intermediários, o pacote é recebido inteiramente no roteador intermediário e é armazenado em uma área (buffer) até que uma linha de transmissão esteja disponível para a sua retransmissão. A escolha do caminho é determinada, pelo algoritmo de roteamento, em função das condições operacionais das linhas de transmissão (disponibilidade das linhas, banda utilizada em cada linha, etc.).

Um projetista de uma rede WAN precisa especificar tanto a velocidade das linhas de transmissão quanto à capacidade dos roteadores que serão instalados, notadamente, a velocidade de roteamento e a capacidade de armazenamento de pacotes (tamanho dos buffers). Caso as linhas de transmissão sejam especificadas abaixo do valor necessário, os usuários sentirão, inevitavelmente, problemas de lentidão. Se os roteadores não forem especificados corretamente, não só a velocidade de acesso às informações sofrerá impacto, mas também poderão ocorrer perdas de pacotes, em virtude do pouco espaço (tamanho dos buffers) definido para armazenamento temporário nos roteadores. Hacker & Smith (2008) justificam a necessidade do desenvolvimento de modelos de simulação para a avaliação de desempenho de WANs visto que i) testes podem prejudicar tráfego crítico na rede, ii) é impraticável separar a rede em segmentos para experimentos e testes de desempenho, iii) não é fácil reproduzir uma situação real de tráfego em uma rede desta dimensão e iv) não é possível configurar os sistemas e a rede para os testes.

Luo & Marin (2005) usam uma mistura de distribuições com caudas acentuadas a fim de modelar o tráfego de protocolos de rede Internet (FTP, HTTP, SMTP, POP e SSH). Entretanto, os próprios autores reconhecem que esta abordagem possui a limitação de que a técnica é específica aos protocolos em questão e que diferentes modelos devem ser construídos para outras aplicações Internet.

O método proposto no item 3.2 pode ser usado para gerar os pacotes necessários, em um ambiente de simulação, de um roteador de origem para um roteador de destino. A complexidade para a geração dos pacotes, supondo que haja n roteadores de origem e destino, é de (n^2-n), visto que cada roteador deve gerar pacotes para os demais (n-1) roteadores. Por outro lado, não é necessário que o algoritmo de geração de pacotes se preocupe com o espaço

disponível nos buffers dos roteadores, pois um dos parâmetros que se busca analisar nestes sistemas é, justamente, a quantidade de pacotes que foram perdidos em função da falta de espaço provocada pelo subdimensionamento da capacidade dos roteadores de tratamento dos pacotes.

Diferentemente do modelo apresentado em 3.1, cada entidade gerada (pacotes) possui um determinado tamanho dependendo do tipo de protocolo e das diversas aplicações existentes nas redes LANs. O tamanho do pacote, nesse caso, será estabelecido aleatoriamente de acordo com a função Φ_2. (YEGENOGLU, F.; FARIS, F.; QADAN, 2000, p. 169) justifica que o tráfego na *Web* (observe-se que a *Web* também é uma WAN) apresenta tamanhos de arquivo de acordo com uma fdp com caudas pesadas (*heavy tailed*), tal como a distribuição de Pareto. Outros autores, Addie, Neame & Zukerman (2002), consideram que o processo PPBP (Poisson Pareto Burst Process) possui propriedades que modelam, de forma consistente o tráfego na Internet. Os processos PPBP são to tipo M/P/∞, sendo processos de chegada Poisson e com tempo de serviço aderentes à distribuição Pareto.

Uma distribuição de probabilidade é dita ser de cauda pesada se suas caudas não são exponencialmente limitadas. Dito de outra forma, as caudas são mais pesadas que a distribuição exponencial, isto é, seja uma variável aleatória *X*, com função cumulativa *F*, essa é dita ser de cauda direita pesada se $\lim_{x\to\infty} e^{\lambda x} \Pr[X > x] = \infty$. A Figura 4.5 mostra uma comparação entre o gráfico da fdp da exponencial com o da Pareto. Observa-se que para os valores iniciais de *x* a probabilidade da distribuição de Pareto é menor que a exponencial. Esse fato é compensado pela probabilidade maior na cauda à direita.

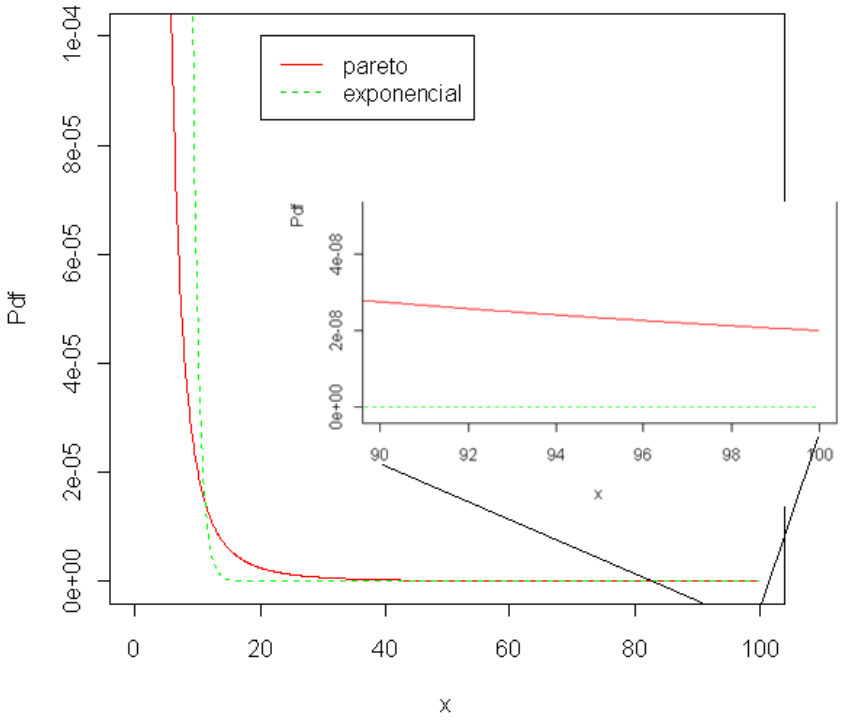

Figura 4.5 – Comparação entre a fdp exponencial e Pareto

Considerando a topologia de rede apresentada em Figura 4.4, os dados de entrada para o modelo são apresentados a seguir:

- $\boldsymbol{P} = 3$;
- $\boldsymbol{C} = 5$;
- $\boldsymbol{NC_c}$ = { ‘Roteador A’, ‘Roteador B’, ‘Roteador C’, ‘Roteador D’, ‘Roteador E’, }
- $\boldsymbol{B_p} = \{0, 30, 60, 90\}$;
- $\boldsymbol{\Phi_1} = -ln(u)$;
- $\boldsymbol{\Phi_2} = \frac{500}{u^{\frac{1}{2}}}$;

- ***Rclo,cld,p =***

	Períodos									
	1					2				
Classes	Serv A	Serv B	Serv C	Serv D	Serv E	Serv A	Serv B	Serv C	Serv D	Serv E
Serv A	0	50	90	30	70	0	50	90	30	70
Serv B	10	0	100	40	80	10	0	100	40	80
Serv C	20	60	0	0	90	20	60	0	0	90
Serv D	30	70	10	50	100	30	70	10	50	100
Serv E	40	80	20	60	0	40	80	20	60	0

	Períodos				
	3				
Classes	Serv A	Serv B	Serv C	Serv D	Serv E
Serv A	0	55	95	135	175
Serv B	15	0	105	145	185
Serv C	25	65	0	155	195
Serv D	35	75	115	0	205
Serv E	45	85	125	165	0

Para estes dados de entrada, tem-se então um processo PPBP. De acordo com $R_{clo,cld,p}$, deverão ser gerados, em média, 155 pacotes do *'Serv C'* para o *'Serv D'* no período 3, com chegadas Poisson**,** conforme estabelecido por Φ_1. O atributo *Capacidade* de cada pacote a ser gerado terá seus valores atribuídos de acordo com uma distribuição Pareto com parâmetros $k = 500$ e $\alpha = 2$. Isto significa que o tamanho médio dos pacotes deverá ser igual a $E[\mathbf{\Phi_2}] = = \frac{\alpha * k}{(\alpha - 1)} = = \frac{2 * 500}{(2 - 1)} = 1000$. Como a distribuição é Pareto, a probabilidade de se ter pacotes grandes não é negligenciável, o que implica a ocorrência de rajadas de transmissão (*bursts*), fenômeno frequente em redes de longa distância

Executou-se uma simulação com 1000 rodadas[10] e os seguintes resultados foram obtidos, conforme ilustra o Quadro 4.3.

[10] O tempo computacional foi 0,95 por rodada.

Totais por classe e 1º período						
Período	clo	cld	Média	HW	Mínimo	Máximo
1	Serv A	Serv A	0,000	0,000	0	0
		Serv B	49,924	0,449	31	70
		Serv C	89,734	0,602	60	122
		Serv D	29,994	0,335	13	47
		Serv E	69,815	0,512	46	100
	Serv B	Serv A	10,069	0,194	2	21
		Serv B	0,000	0,000	0	0
		Serv C	99,722	0,632	69	132
		Serv D	39,918	0,384	20	60
		Serv E	79,546	0,541	51	110
	Serv C	Serv A	20,084	0,275	9	36
		Serv B	59,863	0,490	37	87
		Serv C	0,000	0,000	0	0
		Serv D	0,000	0,000	0	0
		Serv E	89,962	0,615	61	125
	Serv D	Serv A	29,885	0,329	14	51
		Serv B	70,598	0,549	48	103
		Serv C	9,956	0,195	1	20
		Serv D	50,408	0,433	30	71
		Serv E	99,539	0,589	72	133
	Serv E	Serv A	40,057	0,391	22	62
		Serv B	79,851	0,546	57	111
		Serv C	20,110	0,273	4	36
		Serv D	60,016	0,462	37	82
		Serv E	0,000	0,000	0	0

Quadro 4.3 - Totais de eventos gerados (1º período) para a simulação WAN por período com IC em 95% para 1000 rodadas

Deve-se observar que os totais gerados convergem para os valores definidos em $R_{clo,cld,1}$. Por brevidade, apenas os dados do primeiro período são exibidos. Para todos os demais períodos os valores, também, convergiram para os dados de entrada estabelecidos em $R_{clo,cld,p}$. Além disso, o total de eventos gerados em média por rodada, 4400,9 com HW (IC em 95%) igual a 4,2962, também, converge para o valor $\sum_{clo=1}^{C}\sum_{cld=1}^{C}\sum_{p=1}^{P} R_{clo,cld,p} = 4400$. Entre as 1000 rodadas executadas a quantidade mínima e máxima de pacotes foi 4163 e 4614, respectivamente.

Para o atributo *Capacidade* (tamanho dos pacotes) foi obtido o valor médio de 1002,81 com HW igual a 41,78 com tamanho mínimo e máximo de 500 e 43041, respectivamente.

4.3 APLICAÇÃO DOS MÉTODOS NA ÁREA DE LOGÍSTICA

Grande parte dos estudos e projetos desenvolvidos para ambientes portuários trata de problemas relacionados, de alguma forma, à ocorrência de filas (ASPEREN *et al.*, 2003b; HEISEY, 2005; DAI *et al*., 2003). Este fato conduz o pesquisador a considerar, como uma primeira abordagem de estudo, o uso da teoria de filas.

A grande complexidade dos ambientes portuários justifica a aplicação da simulação. Nestes ambientes há uma grande variabilidade no intervalo de chegada de navios e nos tempos de serviços dos navios atracados (BARDOS, 1999; GAMBARDELLA & RIZZOLI, 2000; ASPEREN *et al.*, 2003a). Além disso, restrições nas operações em virtude de interrupções de embarque por motivos climáticos, influência das marés, falhas em equipamentos ou variações em turnos de trabalhadores impossibilitam a análise destes sistemas sem o auxílio de um modelo matemático que considere os fatores aleatórios que permeiam os diversos processos (MASNIK FERREIRA, M. A.; MENDES JÚNIOR, R.; CARNIERI, C., 2007). Apesar de toda esta complexidade, o administrador do sistema necessita de uma técnica de estudo que permita cotejar diferentes políticas antes de serem implantadas, pois o custo de modificações no sistema é elevado e, muitas vezes, estas modificações são irreversíveis.

Masnik Ferreira, Mendes Júnior & Carnieri (2007) mostram a dificuldade de se modelar a chegada de caminhões e vagões para descarga nos silos em ambientes portuários. A chegada destes caminhões e trens se dá tanto por fatores aleatórios quanto determinísticos. O envio de mercadoria ao porto é feito tendo em conta a necessidade futura de embarque de navios. Após estes caminhões e trens terem sido despachados de suas origens para o porto, diversos fatores são passíveis de ocorrer gerando não só atrasos, mas até a não chegada ao porto destes trens e caminhões. Caso fosse utilizada somente uma distribuição de probabilidade para se modelar estas chegadas de mercadorias ao porto, dois fatores indesejáveis poderiam ocorrer na data prevista do embarque dos navios: um volume maior ou menor de mercadoria que o necessário para o embarque do navio. Ressalte-se que a administração dos Terminais Portuários (TPs) - agentes responsáveis pelo armazenamento e embarque das mercadorias no porto -, por meio de um planejamento diário, busca sempre corrigir estas distorções nos volumes já armazenados de mercadorias a fim de que antes até da

data prevista, todo o volume necessário para o embarque esteja no porto. Ademais, o navio só adquire a condição de atracação nos berços caso tenha toda a mercadoria a ser embarcada já armazenada nos silos dos TPs.

4.3.1 Descrição do Complexo de Granel no Porto de Paranaguá

O complexo de granel do porto de Paranaguá é um intrincado sistema composto pelos pátios de triagem de vagões e caminhões, contando com 11 Terminais Portuários (TPs) (APPA silo vertical, APPA silo horizontal, Coamo, Cotriguaçu, Cargill, CBL, Coinbra, Centro Sul, AGTL, Soccepar e Bunge), 5 berços de atracação e um sistema integrado de correias de transmissão que interliga os silos dos TPs aos *"ship-loaders"*. Estes últimos realizam o embarque dos navios, conforme mostrado na Figura 4.6.

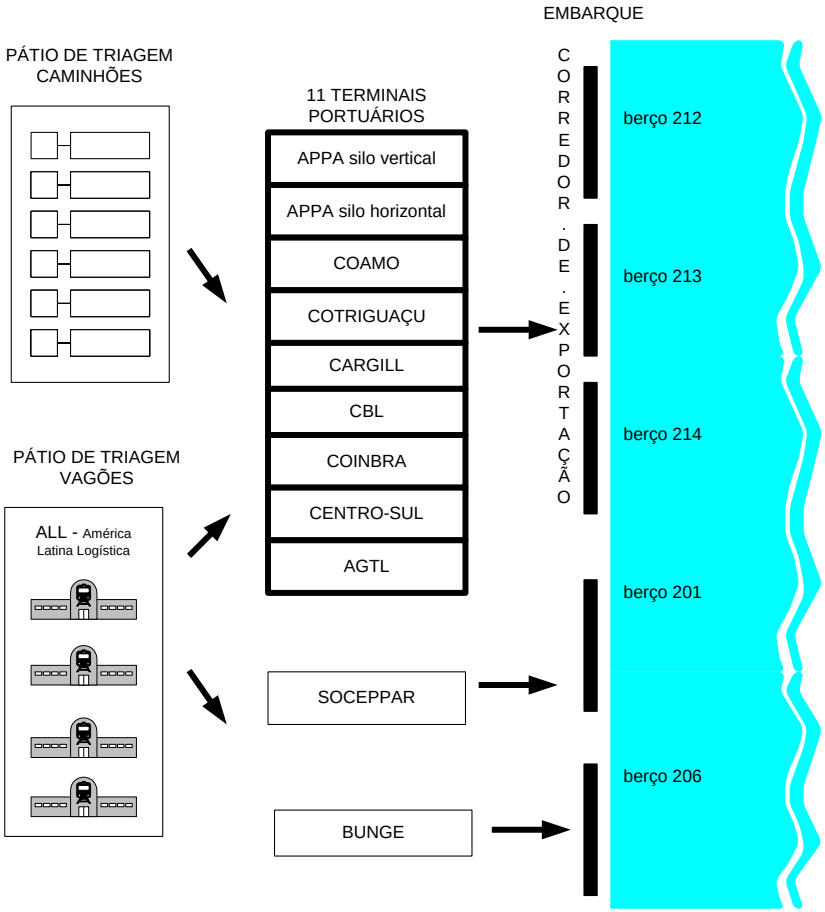

Figura 4.6 – Complexo de granel do porto de Paranaguá**.**

Fonte: (MASNIK FERREIRA; ROSA; CARNIERI, 2006).

Todo o gerenciamento do complexo portuário é feito pela Administração dos Portos de Paranaguá e Antonina – APPA, desde a chegada dos vagões e caminhões aos pátios de triagem até o embarque das mercadorias nos berços de atracação. Além disso, a APPA dispõe de um silo vertical, com capacidade de 96 mil toneladas, destinado ao armazenamento de soja e de 4 silos verticais, com capacidade individual de 12 mil toneladas - totalizando 48 mil toneladas -, destinados ao armazenamento de farelo de soja.

Cada TP, inclusive a APPA, é responsável pela programação de envio ao porto das mercadorias na quantidade e na antecedência necessária a fim de atender à demanda futura e prevista de embarque nos navios. Esta programação é feita de forma isolada e, geralmente, sem o uso de uma ferramenta automatizada, não podendo prescindir, portanto, dos funcionários envolvidos nestas atividades.

O Quadro 4.4 mostra a relação entre os componentes existentes nos sistema portuário descrito acima e os componentes do modelo geral mostrado na Figura 1.2.

Modelo Proposto	**Sistema Portuário**
Entidades	Caminhões e vagões que são enviados ao porto para efetuar a descarga de mercadorias em um TP. Suas cargas são depositadas nos silos (buffers) ocupando um espaço correspondente.
Destino	Terminais Portuários (TPs).
Buffer	Silos onde as mercadorias são armazenadas até o seu embarque nos navios.

Quadro 4.4 Correspondência entre os componentes do sistema portuário ao modelo proposto na Figura 1.2.

4.3.2 Modelo de Simulação Original do Porto de Paranaguá

Masnik Ferreira, Mendes Júnior & Carnieri (2007) apresentam um modelo de simulação para o complexo de carga a granel do Porto de Paranaguá (Figura 4.7), cujo

objetivo foi o de avaliar o comportamento do sistema quando submetido a determinado volume de trabalho. Considerando os objetivos da presente pesquisa, a seguir será mostrado como os autores do trabalho resolveram o problema da geração de entidades nesse modelo de simulação.

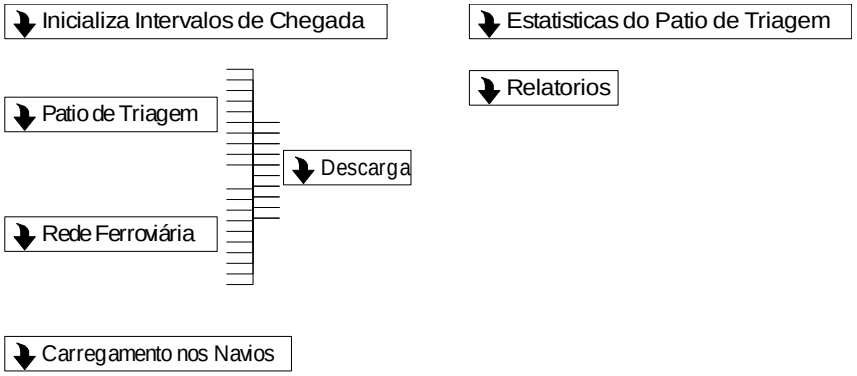

Figura 4.7 – Submodelos que implementam o modelo de simulação do complexo de granel do Porto de Paranaguá

O modelo estabelece um período de 30 dias de trabalho de descarga de caminhões e vagões e embarque de navios. A escolha deste período deveu-se ao fato de que as estatísticas disponíveis das atividades portuárias são fornecidas de forma mensal. O trabalho considerou a possibilidade dos navios embarcarem até 4 produtos simultaneamente (‘soja’, ‘farelo’, ‘milho’ e ‘trigo’) e uma quinta categoria chamada genericamente de ‘outros’.

No modelo de simulação desenvolvido pelos autores, a geração de entidades é feita, inicialmente, por meio de uma planilha, no Microsoft Excel, a qual calcula, de forma estática, as taxas de chegada, por dia, que devem ser observadas na simulação a fim de que os volumes de mercadorias sejam atingidos até a data da chegada dos navios no porto. Este cálculo é feito fixando uma antecedência de 14 dias de modo que, na data prevista, toda a mercadoria já esteja armazenada nos silos dos TPs. Isso significa que as mercadorias começam sempre a serem enviadas ao porto 14 dias antes da data prevista de embarque. Desta forma, se for considerada a primeira data de embarque, estes 14 dias tornam-se o período de aquecimento (*warm-up*) da simulação, visto que o relevante, neste modelo de simulação, é a análise do comportamento do sistema a partir da primeira data de embarque.

O cálculo feito por essa planilha, como mencionam os autores, não é trivial. Além de garantir que na data prevista toda a mercadoria já esteja armazenada nos silos, o cálculo deve

levar em consideração a capacidade de armazenamento dos silos. Iniciar a geração de caminhões e vagões com 14 dias de antecedência é uma medida de segurança que visa garantir a totalidade das mercadorias na data dos embarques. No entanto, caso a quantidade gerada seja superior à capacidade dos silos, já no período de *warm-up*, ocorrerá o enchimento dos silos, provocando filas indesejáveis no entorno dos TPs. Estas filas irão, certamente, distorcer os dados estatísticos relacionados com o desempenho dos TPs, posto que este fenômeno não ocorre no sistema real.

O modelo desenvolvido modifica, por meio de *Visual Basic Application*, diariamente (submodelo '*Inicializa Intervalos de Chegada'*), os intervalos de chegada que devem ser observados a fim de garantir que a quantidade necessária de mercadoria entre no sistema durante um determinado dia. Para cada mercadoria há uma variável que registra o intervalo entre chegadas de caminhão e trem que deve ser observado. Por exemplo, o vetor *intervaloChegadaSoja(TP, meio de transporte)* armazena os intervalos de chegada que devem ser observados para que a quantidade exata chegue a cada TP, por cada meio de transporte (caminhão e trem).

Os submodelos '*Pátio de Triagem'* e '*Rede Ferroviária*' implementam a lógica de recebimentos dos caminhões e vagões nos respectivos pátios de triagem. Dentro de cada um desses dois submodelos, há outros cinco submodelos que realizam a geração das entidades (caminhões e trens), conforme mostra a Figura 4.8.

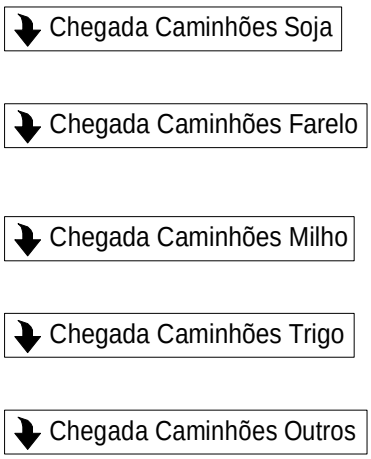

Figura 4.8 – Submodelos que implementam a lógica da entrada de caminhões para cada tipo de mercadoria

Para cada mercadoria, é necessário fazer o tratamento das chegadas referente a cada TP. A Figura 4.9 mostra a lógica de controle para geração de caminhões para o TP 'APPA vertical'. No início da simulação é gerada uma única entidade que governará a sequência de geração de caminhões para esse TP. Inicialmente, é verificado se há 'Soja' programada para ser recebida pelo TP naquele dia. Se não houver, a entidade aguarda 24 horas e volta a fazer o teste no próximo dia. Caso contrário, a entidade aguarda um intervalo definido por *intervaloChegadaSoja(iAPPAv, CAM),* já corretamente inicializado para esse dia. Em seguida, por meio de um módulo '*Separate'* divide-se a entidade em duas. Uma segue adiante no modelo (representando um caminhão de soja destinado ao TP '*APPA vertical'*) e a outra retorna a fim de repetir o procedimento de geração de nova entidade.

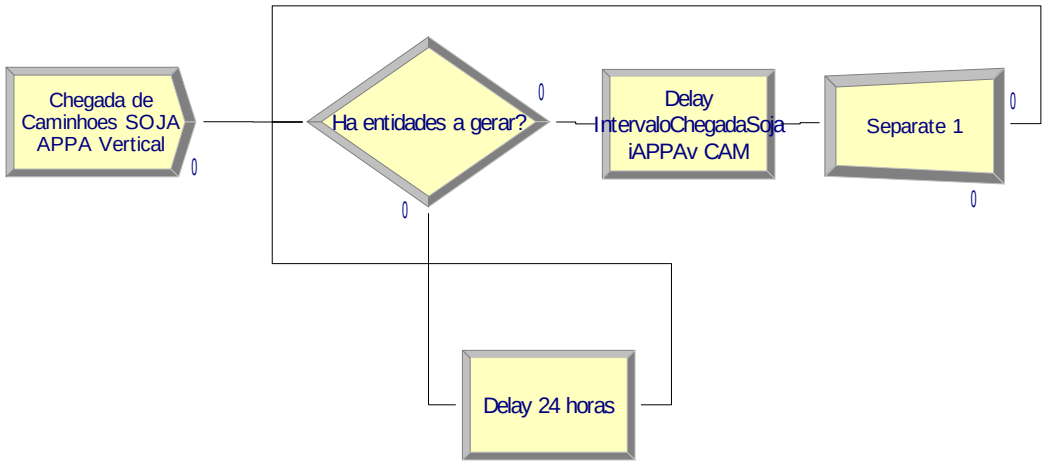

Figura 4.9 – Geração de entidades para um TP '*APPA vertical*' de um determinado meio de transporte (*caminhão*)

Dois aspectos negativos podem ser apontados nessa solução. Primeiro, o modelo apresentado na Figura 4.9 precisa ser replicado para cada mercadoria, TP e meio de transporte. Portanto, como no modelo desenvolvido há 5 classes de mercadorias, 11 TPs e 2 meios de transportes, é preciso replicar 110 modelos semelhantes ao apresentado na Figura 4.9, supondo que todos os TPs operam com todos os tipos de mercadorias. Segundo e, talvez, mais grave ainda, as entidades (caminhões e trens) chegam em intervalos igualmente espaçados – o que evidentemente não ocorre na prática. Qualquer outra forma de tratamento dessas chegadas deverá ser desenvolvida pelo próprio projetista, tendo este que replicar sua solução 110 vezes!

4.3.3 Modelo de Simulação Modificado do Porto de Paranaguá

Usando os métodos propostos para a geração de entidades, a Figura 4.10 mostra como fica o modelo de simulação do complexo do Porto de Paranaguá.

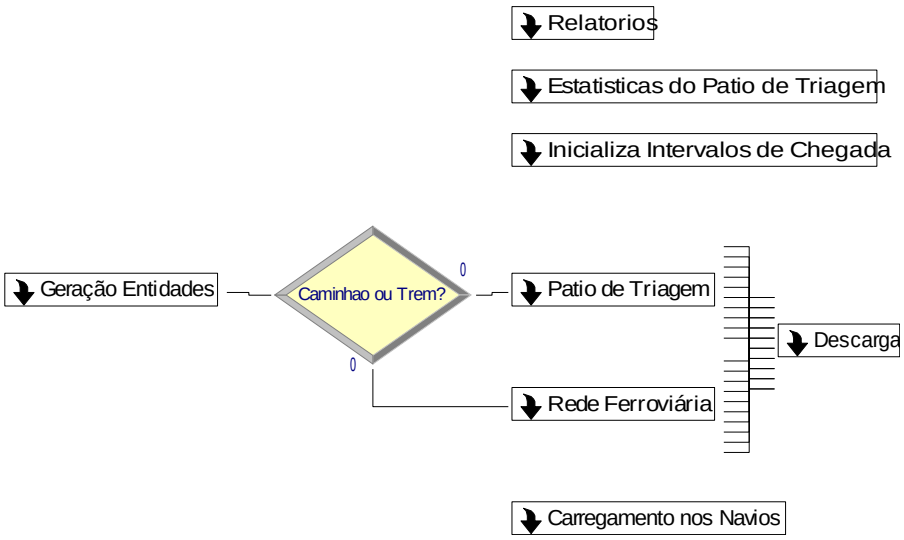

Figura 4.10 – Modelo de simulação do complexo de granel do Porto de Paranaguá modificado

Toda complexidade da geração está contida no submodelo ou *template* (dependendo da implementação feita no *Arena*), liberando o projetista para se concentrar nas questões intrínsecas do modelo.

Foi retirada, dos submodelos '*Pátio de Triagem'* e '*Rede Ferroviária*', a lógica de entrada de entidades apresentadas na Figura 4.8 e na Figura 4.9, ficando apenas com a lógica do recebimento dos caminhões e vagões. Além disso, não foi mais necessário usar a já referida planilha que realizava os cálculos necessários para a geração de caminhões e vagões.

Outra diferença em relação ao modelo original é a sua escalabilidade (capacidade de crescimento do modelo), pois nada precisa ser modificado na geração de entidades (do ponto de vista da modelagem) ao se ampliar a quantidade de TPs, classes (mercadorias) ou meios de transporte.

4.3.4 Exemplo numérico

Considera-se, a seguir, um exemplo numérico que retrata uma demanda mensal, no Porto de Paranaguá, necessária para a realização de embarques de navios no complexo de granel. Desta forma, *PD* = 30 e considera-se que ainda se dispõem de 14 dias para o início dos embarques (*PA* = 10). O vetor de pesos utilizado para o planejamento da programação foi (*p1* = 1, *p2* = 2, *p3* = 3, *p4* = 4) O Quadro 4.6 apresenta os dados de entrada para cada TP e o Quadro 4.7 mostra os dados de demanda (soja e farelo), por período de demanda. O total de demanda para todos os TPs e classes de produto é de 1.930.000 toneladas. Pelos dados de entrada, pode-se observar que o TP '*APPA SV*' opera somente com '*soja'* e o TP '*APPA SH*' opera somente com '*farelo'*. Verifica-se, também, que as taxas para cada meio de transporte variam de acordo com o TP. Por exemplo, o TP '*Coamo'* opera com 75% e 25% por meio de caminhão e trem, respectivamente. O Quadro 4.5 mostra os tempos computacionais para a geração de entidades para cada TP e o tempo total para a geração de todos os TPs.

O Continuação Quadro sumariza os resultados da geração de entidades para esses dados de entrada. Verifica-se que foi gerado 1.931.472 toneladas de mercadorias, muito próximo do valor demandado. Também, observa-se que, para cada demanda, foram geradas as quantidades de meios de transporte (de acordo com o percentual estabelecido) necessárias para atendê-las.

1	APPA Vertical	44s
2	APPA Horizontal	11s
3	Cargill	26s
4	Centro Sul	7s
5	Cotriguaçu	20s
6	Coamo	41s
7	CBL	1m17s
8	Coinbra	36s
9	AGTL	46s
10	Soccepar	12s
11	Bunge	17s
	Total	5m38s

Quadro 4.5 – Tempo Computacional para Cada TP e Tempo Total

AGTL				
	CE	156.000		
	CT	10.000		
	PS	2		
	LSMin	1.000		
	LSMax	5.000		
	C	2		
	NC	{soja, farelo}		
	QMT	2		
	NMT	{caminhão, trem}		
	CMT	{36, 324}		
	PMT		soja	farelo
		caminhão	70%	30%
		Trem	30%	70%

APPA SH				
	CE	48.000		
	CT	10.000		
	PS	2		
	LSMin	1.000		
	LSMax	5.000		
	C	1		
	NC	{farelo}		
	QMT	2		
	NMT	{caminhão, trem}		
	CMT	{36, 324}		
	PMT		farelo	
		caminhão	30%	
		Trem	70%	

APPA SV				
	CE	96.000		
	CT	13.000		
	PS	2		
	LSMin	300		
	LSMax	3.000		
	C	1		
	NC	{soja}		
	QMT	2		
	NMT	{caminhão, trem}		
	CMT	{36, 324}		
	PMT		soja	
		caminhão	70%	
		Trem	30%	

Bunge				
	CE	96.000		
	CT	10.000		
	PS	2		
	LSMin	1.000		
	LSMax	5.000		
	C	2		
	NC	{soja, farelo}		
	QMT	2		
	NMT	{caminhão, trem}		
	CMT	{36, 324}		
	PMT		soja	farelo
		caminhão	10%	0%
		Trem	90%	100%

Cargil				
	CE	115.000		
	CT	10.000		
	PS	2		
	LSMin	1.000		
	LSMax	5.000		
	C	2		
	NC	{soja, farelo}		
	QMT	2		
	NMT	{caminhão, trem}		
	CMT	{36, 324}		
	PMT		soja	farelo
		caminhão	70%	30%
		Trem	30%	70%

CBL				
	CE	115.000		
	CT	10.000		
	PS	2		
	LSMin	1.000		
	LSMax	5.000		
	C	2		
	NC	{soja, farelo}		
	QMT	2		
	NMT	{caminhão, trem}		
	CMT	{36, 324}		
	PMT		soja	farelo
		caminhão	10%	10%
		Trem	90%	90%

Centro Sul				
	CE	70.000		
	CT	10.000		
	PS	2		
	LSMin	1.000		
	LSMax	5.000		
	C	2		
	NC	{soja, farelo}		
	QMT	2		
	NMT	{caminhão, trem}		
	CMT	{36, 324}		
	PMT		soja	farelo
		caminhão	70%	30%
		Trem	30%	70%

Coamo				
	CE	90.000		
	CT	10.000		
	PS	2		
	LSMin	1.000		
	LSMax	5.000		
	C	2		
	NC	{soja, farelo}		
	QMT	2		
	NMT	{caminhão, trem}		
	CMT	{36, 324}		
	PMT		soja	farelo
		caminhão	75%	75%
		Trem	25%	25%

Coinbra				
	CE	55.000		
	CT	10.000		
	PS	2		
	LSMin	1.000		
	LSMax	5.000		
	C	2		
	NC	{soja, farelo}		
	QMT	2		
	NMT	{caminhão, trem}		
	CMT	{36, 324}		
	PMT		soja	farelo
		caminhão	10%	10%
		Trem	90%	90%

Cotriguaçu				
	CE	150.000		
	CT	10.000		
	PS	2		
	LSMin	1.000		
	LSMax	5.000		
	C	1		
	NC	{soja}		
	QMT	2		
	NMT	{caminhão, trem}		
	CMT	{36, 324}		
	PMT		soja	
		caminhão	70%	
		Trem	30%	

Socepar				
	CE	192.000		
	CT	10.000		
	PS	2		
	LSMin	1.000		
	LSMax	5.000		
	C	2		
	NC	{soja, farelo}		
	QMT	2		
	NMT	{caminhão, trem}		
	CMT	{36, 324}		
	PMT		soja	farelo
		caminhão	80%	10%
		Trem	20%	90%

Quadro 4.6 - Dados de entrada para cada TP

PD	Classe	AGTL	APPA SH	APPA SV	Bunge	Cargill	CBL	Centro Sul	Coamo	Coinbra	Cotriguaçu	Soccepar
1	Soja	-	-	30.000	-	10.000	10.000	-	-	-	10.000	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
2	Soja	20.000	-	20.000	-	-	-	-	-	20.000	-	-
	Farelo	-	35.000	-	-	-	-	10.000	20.000	-	-	-
3	Soja	-	-	30.000	-	20.000	-	-	20.000	-	10.000	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
4	Soja	20.000	-	-	35.000	-	20.000	5.000	-	-	-	-
	Farelo	-	-	-	35.000	-	-	-	-	-	-	-
5	Soja	-	-	40.000	-	-	-	-	-	-	10.000	-
	Farelo	-	5.000	-	-	40.000	-	-	-	20.000	-	-
6	Soja	-	-	-	-	-	-	-	-	-	-	50.000
	Farelo	-	-	-	-	-	-	-	-	-	-	-
7	Soja	5.000	-	35.000	-	-	5.000	5.000	-	-	5.000	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
8	Soja	-	-	-	-	-	-	-	-	-	-	-
	Farelo	-	-	-	-	-	-	-	-	-	-	40.000
9	Soja	10.000	-	10.000	-	20.000	20.000	5.000	20.000	-	15.000	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
10	Soja	-	-	40.000	-	-	-	-	-	-	10.000	60.000
	Farelo	20.000	15.000	-	-	-	5.000	5.000	-	-	-	-
11	Soja	-	-	30.000	-	20.000	-	-	-	-	10.000	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
12	Soja	20.000	-	-	-	-	20.000	10.000	20.000	30.000	-	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
13	Soja	-	-	10.000	-	-	-	-	-	30.000	20.000	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
14	Soja											
	Farelo											
15	Soja	20.000	-	30.000	-	20.000	10.000	-	10.000	-	-	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
16	Soja	-	-	-	-	-	-	-	-	-	-	-
	Farelo	20.000	20.000	-	-	-	10.000	20.000	-	20.000	-	-
17	Soja											
	Farelo											
18	Soja	-	-	20.000	35.000	10.000	-	-	-	-	30.000	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
19	Soja	-	-	25.000	-	-	-	-	20.000	-	-	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
20	Soja	-	-	-	-	-	-	-	-	-	-	-
	Farelo	-	-	-	-	-	-	-	-	-	-	30.000
21	Soja	20.000	-	-	-	-	20.000	10.000	-	-	-	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
22	Soja	10.000	-	40.000	-	-	-	-	-	-	10.000	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
23	Soja	10.000	-	-	-	20.000	-	5.000	-	-	5.000	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
24	Soja	20.000	-	40.000	-	-	20.000	-	10.000	10.000	-	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
25	Soja	-	-	30.000	35.000	10.000	10.000	-	-	-	-	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
26	Soja	-	-	-	-	-	-	-	-	-	-	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
27	Soja	-	-	40.000	35.000	-	-	-	20.000	-	-	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
28	Soja	-	-	-	35.000	-	-	-	-	-	-	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
29	Soja	-	-	-	-	-	-	-	-	-	-	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
30	Soja	-	-	-	-	-	-	-	-	-	-	-
	Farelo	-	-	-	-	-	-	-	-	-	-	-
Total		195.000	75.000	470.000	210.000	170.000	150.000	75.000	140.000	130.000	135.000	180.000

Quadro 4.7 - Dados de demanda por classe de produto (soja e farelo) para cada TP

		Farelo		Farelo Total	Qtde Total	Soja		Soja Total	Qtde Total	Total geral Meios Transporte	Qtde Total
Terminal Portuário	**PD**	**caminhão**	**Trem**	**Caminhão**	**Trem**	**caminhão**	**Trem**				
AGTL	2	-	-	-	-	385	19	404	20.016	404	20.016
	4	-	-	-	-	385	19	404	20.016	404	20.016
	7	-	-	-	-	94	5	99	5.004	99	5.004
	9	-	-	-	-	197	9	206	10.008	206	10.008
	10	169	43	212	20.016	-	-	-	-	212	20.016
	12	-	-	-	-	385	19	404	20.016	404	20.016
	15	-	-	-	-	385	19	404	20.016	404	20.016
	16	169	43	212	20.016	-	-	-	-	212	20.016
	21	-	-	-	-	385	19	404	20.016	404	20.016
	22	-	-	-	-	197	9	206	10.008	206	10.008
	23	-	-	-	-	197	9	206	10.008	206	10.008
	24	-	-	-	-	385	19	404	20.016	404	20.016
AGTL Total		338	86	424	40.032	2.995	146	3.141	155.124	3.565	195.156
APPA Horizontal	2	289	76	365	35.028	-	-	-	-	365	35.028
	5	40	11	51	5.004	-	-	-	-	51	5.004
	10	129	32	161	15.012	-	-	-	-	161	15.012
	16	169	43	212	20.016	-	-	-	-	212	20.016
APPA Horizontal Total		627	162	789	75.060	-	-	-	-	789	75.060
APPA Vertical	1	-	-	-	-	582	28	610	30.024	610	30.024
	2	-	-	-	-	385	19	404	20.016	404	20.016
	3	-	-	-	-	582	28	610	30.024	610	30.024
	5	-	-	-	-	779	37	816	40.032	816	40.032
	7	-	-	-	-	685	32	717	35.028	717	35.028
	9	-	-	-	-	197	9	206	10.008	206	10.008
	10	-	-	-	-	779	37	816	40.032	816	40.032
	11	-	-	-	-	582	28	610	30.024	610	30.024
	13	-	-	-	-	197	9	206	10.008	206	10.008
	15	-	-	-	-	582	28	610	30.024	610	30.024
	18	-	-	-	-	385	19	404	20.016	404	20.016
	19	-	-	-	-	488	23	511	25.020	511	25.020
	22	-	-	-	-	779	37	816	40.032	816	40.032
	24	-	-	-	-	779	37	816	40.032	816	40.032
	25	-	-	-	-	582	28	610	30.024	610	30.024
	27	-	-	-	-	779	37	816	40.032	816	40.032
APPA Vertical Total		-	-	-	-	9.142	436	9.578	470.376	9.578	470.376
Bunge	4	1	108	109	35.028	100	97	197	35.028	306	70.056
	18	-	-	-	-	100	97	197	35.028	197	35.028
	25	-	-	-	-	100	97	197	35.028	197	35.028
	27	-	-	-	-	100	97	197	35.028	197	35.028
	28	-	-	-	-	100	97	197	35.028	197	35.028
Bunge Total		1	108	109	35.028	500	485	985	175.140	1.094	210.168
Cargill	1	-	-	-	-	197	9	206	10.008	206	10.008
	3	-	-	-	-	385	19	404	20.016	404	20.016
	5	338	86	424	40.032	-	-	-	-	424	40.032
	9	-	-	-	-	385	19	404	20.016	404	20.016
	11	-	-	-	-	385	19	404	20.016	404	20.016
	15	-	-	-	-	385	19	404	20.016	404	20.016
	18	-	-	-	-	197	9	206	10.008	206	10.008
	23	-	-	-	-	385	19	404	20.016	404	20.016
	25	-	-	-	-	197	9	206	10.008	206	10.008
Cargill Total		338	86	424	40.032	2.516	122	2.638	130.104	3.062	170.136
CBL	1	-	-	-	-	26	28	54	10.008	54	10.008
	4	-	-	-	-	52	56	108	20.016	108	20.016
	7	-	-	-	-	13	14	27	5.004	27	5.004
	9	-	-	-	-	52	56	108	20.016	108	20.016
	10	13	14	27	5.004	-	-	-	-	27	5.004
	12	-	-	-	-	52	56	108	20.016	108	20.016
	15	-	-	-	-	26	28	54	10.008	54	10.008
	16	26	28	54	10.008	-	-	-	-	54	10.008
	21	-	-	-	-	52	56	108	20.016	108	20.016
	24	-	-	-	-	52	56	108	20.016	108	20.016
	25	-	-	-	-	26	28	54	10.008	54	10.008
CBL Total		39	42	81	15.012	351	378	729	135.108	810	150.120

Quadro 4.8 – Resultados da geração de entidades (caminhões e trens) para cada TP

Centro Sul	2	80	22	102	10.008	-	-	-	-	102	10.008
	4	-	-	-	-	94	5	99	5.004	99	5.004
	7	-	-	-	-	94	5	99	5.004	99	5.004
	9	-	-	-	-	94	5	99	5.004	99	5.004
	10	40	11	51	5.004	-	-	-	-	51	5.004
	12	-	-	-	-	197	9	206	10.008	206	10.008
	16	169	43	212	20.016	-	-	-	-	212	20.016
	21	-	-	-	-	197	9	206	10.008	206	10.008
	23	-	-	-	-	94	5	99	5.004	99	5.004
Centro Sul Total		289	76	365	35.028	770	38	808	40.032	1.173	75.060
Coamo	2	421	15	436	20.016	-	-	-	-	436	20.016
	3	-	-	-	-	421	15	436	20.016	436	20.016
	9	-	-	-	-	421	15	436	20.016	436	20.016
	12	-	-	-	-	421	15	436	20.016	436	20.016
	15	-	-	-	-	206	8	214	10.008	214	10.008
	19	-	-	-	-	421	15	436	20.016	436	20.016
	24	-	-	-	-	206	8	214	10.008	214	10.008
	27	-	-	-	-	421	15	436	20.016	436	20.016
Coamo Total		421	15	436	20.016	2.517	91	2.608	120.096	3.044	140.112
Coinbra	2	-	-	-	-	52	56	108	20.016	108	20.016
	5	52	56	108	20.016	-	-	-	-	108	20.016
	12	-	-	-	-	87	83	170	30.024	170	30.024
	13	-	-	-	-	87	83	170	30.024	170	30.024
	16	52	56	108	20.016	-	-	-	-	108	20.016
	24	-	-	-	-	26	28	54	10.008	54	10.008
Coinbra Total		104	112	216	40.032	252	250	502	90.072	718	130.104
Cotriguaçu	1	-	-	-	-	197	9	206	10.008	206	10.008
	3	-	-	-	-	197	9	206	10.008	206	10.008
	5	-	-	-	-	197	9	206	10.008	206	10.008
	7	-	-	-	-	94	5	99	5.004	99	5.004
	9	-	-	-	-	291	14	305	15.012	305	15.012
	10	-	-	-	-	197	9	206	10.008	206	10.008
	11	-	-	-	-	197	9	206	10.008	206	10.008
	13	-	-	-	-	385	19	404	20.016	404	20.016
	18	-	-	-	-	582	28	610	30.024	610	30.024
	22	-	-	-	-	197	9	206	10.008	206	10.008
	23	-	-	-	-	94	5	99	5.004	99	5.004
Cotriguaçu Total		-	-	-	-	2.628	125	2.753	135.108	2.753	135.108
Soccepar	6	-	-	-	-	1.110	31	1.141	50.004	1.141	50.004
	8	113	111	224	40.032	-	-	-	-	224	40.032
	10	-	-	-	-	1.334	37	1.371	60.012	1.371	60.012
	20	87	83	170	30.024	-	-	-	-	170	30.024
Soccepar Total		200	194	394	70.056	2.444	68	2.512	110.016	2.906	180.072
Total geral		2.357	881	3.238	370.296	24.115	2.139	26.254	1.561.176	29.492	1.931.472

Continuação Quadro 4.8 – Resultados da geração de entidades (caminhões e trens) para cada TP

Ao todo foram produzidos 29.493 eventos de geração de entidades, portanto, não é razoável a sua listagem completa. Para fins de exemplificação, o Quadro 4.9 lista os eventos gerados a fim de que se atenda a necessidade do período de demanda (13) do TP *´APPA SV*' relativa a 10.000 ton de soja. A programação foi iniciada no período 6 e finalizada no período 10, respeitando 2 períodos de segurança até o período previsto para a chegada do navio. Se forem somadas as quantidades de mercadorias trazidas por esta sequência de chegadas, verifica-se que foram geradas 10.008 ton.

P	Hora Chegada	Meio Transporte	P	Hora Chegada	Meio Transporte	P	Hora Chegada	Meio Transporte	P	Hora Chegada	Meio Transporte	P	Hora Chegada	Meio Transporte	P	Hora Chegada	Meio Transporte
6	457,41	caminhão	8	511,74	caminhão	9	533,30	caminhão	9	548,88	caminhão	10	562,55	caminhão	10	572,21	caminhão
	458,85	caminhão		512,16	caminhão		535,54	caminhão		549,10	caminhão		562,79	caminhão		572,53	caminhão
	458,91	caminhão		513,30	caminhão		536,09	caminhão		549,23	Trem		563,61	caminhão		573,29	caminhão
	459,04	caminhão		513,49	caminhão		536,89	caminhão		549,81	caminhão		563,92	caminhão		573,32	caminhão
	460,16	caminhão		514,01	caminhão		536,95	caminhão		551,03	caminhão		564,13	caminhão		573,74	caminhão
	462,31	Trem		514,85	caminhão		537,35	caminhão		551,75	caminhão		564,50	caminhão		574,07	caminhão
	465,65	caminhão		515,49	caminhão		537,46	caminhão		552,00	caminhão		564,74	caminhão		574,15	caminhão
	467,20	caminhão		517,70	caminhão		537,48	caminhão		552,00	Trem		564,79	caminhão		574,53	caminhão
	467,35	caminhão		519,05	caminhão		537,56	caminhão	10	552,24	caminhão		565,01	caminhão		574,85	caminhão
	467,75	caminhão		519,17	caminhão		537,74	caminhão		552,39	caminhão		565,06	caminhão		574,86	caminhão
	468,67	caminhão		520,35	caminhão		537,85	caminhão		552,45	caminhão		565,49	caminhão		574,87	caminhão
	468,89	caminhão		520,85	caminhão		537,90	caminhão		552,51	caminhão		565,55	caminhão		574,93	caminhão
	469,46	caminhão		521,08	caminhão		538,17	caminhão		552,53	caminhão		565,62	caminhão		575,66	caminhão
	470,10	Trem		521,26	caminhão		538,76	caminhão		552,73	caminhão		565,86	caminhão		575,82	caminhão
	470,16	caminhão		521,45	caminhão		539,46	caminhão		552,79	caminhão		565,87	caminhão		575,85	caminhão
	470,24	caminhão		521,48	caminhão		540,11	Trem		553,65	caminhão		566,11	caminhão		576,00	caminhão
	470,57	caminhão		521,69	caminhão		540,15	caminhão		553,75	caminhão		566,35	caminhão			
	472,65	caminhão		522,21	caminhão		541,07	caminhão		554,60	caminhão		566,51	caminhão			
	475,22	caminhão		522,47	caminhão		541,68	caminhão		554,65	caminhão		566,56	caminhão			
	475,41	caminhão		522,98	caminhão		542,37	caminhão		554,72	caminhão		567,09	caminhão			
	476,40	caminhão		523,25	caminhão		542,66	caminhão		555,01	caminhão		567,35	caminhão			
	480,26	caminhão		523,63	caminhão		542,96	caminhão		556,09	caminhão		567,67	caminhão			
7	484,04	Trem		523,77	caminhão		543,27	caminhão	10	556,32	caminhão		567,73	caminhão			
	489,84	Trem		523,84	caminhão		543,55	caminhão		556,40	caminhão		567,80	caminhão			
	497,97	Trem		525,70	caminhão		543,71	caminhão		556,57	caminhão		567,81	caminhão			
	504,00	Trem	9	528,28	caminhão		543,79	caminhão		556,96	caminhão		568,28	caminhão			
8	504,60	caminhão		528,46	caminhão		544,77	caminhão		557,09	caminhão		568,51	caminhão			
	505,31	caminhão		528,90	caminhão		545,84	caminhão		558,40	caminhão		568,63	caminhão			
	505,77	caminhão		528,94	caminhão		545,85	caminhão		558,76	caminhão		568,68	caminhão			
	505,90	caminhão		529,30	caminhão		546,42	caminhão		558,89	caminhão		568,82	caminhão			
	506,64	caminhão		529,40	caminhão		547,16	caminhão		559,59	caminhão		569,05	caminhão			
	506,99	caminhão		529,61	caminhão		547,17	caminhão		560,69	caminhão		569,82	caminhão			
	507,02	caminhão		530,10	caminhão		547,80	caminhão		561,01	caminhão		570,06	caminhão			
	509,22	caminhão		531,70	caminhão		547,80	caminhão		561,15	caminhão		570,24	caminhão			
	510,41	caminhão		531,81	caminhão		547,95	caminhão		561,61	caminhão		571,07	caminhão			
	510,50	caminhão		531,97	caminhão		548,26	caminhão		561,63	caminhão		571,41	caminhão			
	510,57	caminhão		532,39	caminhão		548,49	caminhão		562,16	caminhão		571,52	caminhão			
	510,58	caminhão		533,27	caminhão		548,79	caminhão		562,35	caminhão		571,84	caminhão			

Quadro 4.9 – Eventos gerados para a programação do período 13 da '*APPA SV*'(demanda de 10.000 ton de soja)

5 CONCLUSÃO

Este capítulo foi elaborado tendo em vista os objetivos inicialmente propostos pela pesquisa e sugestões para trabalhos futuros.

5.1 OBJETIVOS DA PESQUISA

Considerando a aplicação dos métodos propostos nas áreas de administração, redes de computadores e logística, conclui-se que os objetivos (geral e específicos) definidos para esta pesquisa foram atingidos.

Conforme exposto na introdução desta pesquisa, a técnica de simulação é adequada para a análise de sistemas complexos. Sendo assim, é de fundamental importância que o ambiente no qual o modelo de simulação será implementado apresente ao projetista um elevado nível de abstração, permitindo que este se concentre exclusivamente na complexidade do sistema. Desta forma, acredita-se que a simulação poderá, cada vez mais, ser considerada como ferramenta de análise de sistemas cuja complexidade seja muito difícil de tratar com o uso de outras técnicas da pesquisa operacional.

Apesar de esta pesquisa ter tido sua origem quando do desenvolvimento de modelos de simulação para ambientes portuários (MASNIK FERREIRA; MENDES JÚNIOR; CARNIERI, 2007), é preciso destacar que há outros sistemas que apresentam características semelhantes, aumentando, desta forma, as possibilidades de aplicação dos métodos propostos. Em empresas florestais, por exemplo, há a necessidade de se transportar toras de diferentes locais (talhões) a fim de atender a uma demanda tanto da indústria madeireira quanto da de papel. O processo de envio de toras é planejado (determinístico), há um espaço limitado nos destinos para receber estas toras e há diversas ocorrências aleatórias que permeiam este processo, como quebras de caminhões e chuvas que interrompem as estradas de acesso a muitos talhões.

Dentre os três métodos apresentados no capítulo 3, evidentemente, o mais complexo e, talvez o mais importante do ponto de vista de sua aplicabilidade, é o método para *Processos*

DE RENOVAÇÃO com taxas definidas por meio de planejamento. Os métodos de geração de entidades propostos podem ser implementados de forma semelhante com a *amostragem descritiva*. A ideia é a de separar o algoritmo apresentado em 3.3.3 em duas etapas: no início da simulação, executa-se o planejamento da produção (determinação dos valores de $VT_{p,d,c}$ e $VTMT_{p,d,c,mt}$) e , no início de cada rodada da simulação, executa-se o procedimento *DirectAlgorithmTaxasCalculadas* a fim de se gerar os eventos de chegada, garantindo-se a variabilidade nas sequências de chegada. Desta forma, uma simulação completa propiciaria a análise dos efeitos de um planejamento de programação específico durante todas as rodadas. Modificando-se o vetor de pesos (p_i), diferentes alternativas de planejamento podem ser simuladas, pois as taxas de chegada ($VTMT_{p,d,c,mt}$) são calculadas em função desses pesos.

O uso da função Φ é um fator chave no modelo proposto, pois a possibilidade da definição pelo projetista da função de distribuição que melhor se ajuste ao processo de chegada, indubitavelmente expande os horizontes de aplicação dos métodos. Para as distribuições que não tenham uma forma inversa de sua função cumulativa, outros métodos apresentados no referencial teórico podem ser usados.

Por fim, o referencial teórico apresentado neste trabalho permite ao projetista implantar os métodos propostos não exclusivamente em pacotes de simulação.

5.2 SUGESTÕES PARA TRABALHOS FUTUROS

Law & Kelton (2000, p. 279) sustentam que não há uma estratégia completa e definitiva para a validação de um modelo para um determinado sistema. Se existisse, argumentam os autores, não haveria a necessidade do desenvolvimento do próprio modelo de simulação. Outros autores, Robinson *et al.* (2004, p. 482) argumentam que apesar de ser possível a aplicação de diversos testes a fim de garantir a validade e a credibilidade de um modelo de simulação, nenhum teste pode assegurar que um modelo é completamente válido. Esses testes fazem parte de um processo no qual a credibilidade do modelo é gradativamente demonstrada. À medida que passa por testes mais rigorosos, a credibilidade do modelo aumenta. Entretanto, nada impede que num teste futuro o modelo falhe devido ao fato de ter sido usado de uma maneira não inicialmente prevista ou inapropriada. Mesmo assim, é importante destacar que um modelo pode ser útil independentemente de estar fora de alcance

sua completa e rigorosa validação. É dentro desta perspectiva que se sugerem trabalhos futuros.

É de fundamental importância para que se consiga maior abrangência e, também, aprimoramento dos métodos propostos desta pesquisa que, em primeiro lugar, busque-se a aplicação desses métodos em uma grande variedade de sistemas reais. O exemplo já citado na própria conclusão - indústria madeira e de papel - não é único. Considere-se, por exemplo, um modelo de simulação cujo objetivo seja o de avaliar a adequação ou não de um aeroporto tanto de seu estacionamento, áreas internas e quantidade de atendentes nas áreas de *check-in*. A quantidade de pessoas que chegam a um aeroporto, apesar de certa aleatoriedade no instante de chegada, depende exclusivamente da programação de chegadas e partidas de aviões (aplicação do modelo apresentado no item 3.3). Dessa ampla aplicação dos métodos é que não só sua validade e credibilidade, mas, fundamentalmente, sua utilidade pode ser demonstrada.

Ressalte-se, ainda, que em função dos resultados obtidos pela aplicação do modelo apresentado no item 3.3, em sistemas portuários, não se verificou a necessidade de efetuar a normalização do vetor de pesos, o que pode vir a ser necessário quando da sua aplicação em outros sistemas reais.

Outra possibilidade de pesquisa, no modelo apresentado no item 3.3, é a adoção de metaheurísticas a fim de resolver o problema do planejamento da programação (obtenção dos valores de $VT_{p,d,c}$, posteriormente convertidos para $VTMT_{p,d,c,mt}$). Isso permitirá a comparação com os resultados obtidos pelo PPLIM (apresentado no item 3.3.3.1) e, consequentemente, dispensará o uso de um pacote específico para a solução do PPLIM.

REFERÊNCIAS

ADDIE, R. G.; NEAME, T. D.; ZUKERMAN, M. **Performance evaluation of a queue fed by a poisson pareto burst process**," Computer Networks, Vol. 40, Nº. 3, October, 2002, p. 377–397.

ASPEREN, E. V.; DEKKER, R.; POLMAN, M.; ARONS, H. S. **Modeling Ship Arrivals in Ports**. Proceedings of the 2003 Winter Simulation Conference, USA, 2003a.

ASPEREN, E. V.; DEKKER, R.; POLMAN, M.; ARONS, H. S. **Allocation of Ships in a Port Simulation**. 15th European Simulation Symposium and Exhibition, Delft, Netherlands, 2003b.

BALCI, O. **"Verification, Validation, and Testing"**, In: **Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice**. J. Banks, John Wiley & Sons, New York, 1998, p. 335-393.

BALCI, O. **A Methodology for Certification of Modeling and Simulation Applications**. ACM Transactions on Modeling and Computer Simulation, Vol 11, Nº 4, October, 2001, p 352-377.

BANKS, J. **"Principles of Simulation"**, In: **Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice**. J. Banks, John Wiley & Sons, New York, 1998, p. 3-30.

BARBOZA, A. O.; CARNIERI, C.; STEINER, M. T. A.; SIQUEIRA, P. H. **Técnicas da pesquisa operacional no problema de horários de atendentes em centrais telefônicas**. Gestão & Produção, Vol. 10, Nº 1, Abril, São Carlos, 2003.

BARDOS, Alex. **Application of Computer Simulation for Bulk Port Facility Designs**. Journal Edition 9, Port Technology International, 1999

BARRY, J. R. **Probabilidade, um curso em nível intermediário**. 2ª edição, IMPA (RJ), 1996.

BEICHELT, F. **Stochastic Processes in Science, Engineering, and Finance**. CRC Press, 2006.

BRATLEY, P.; FOX, B. L.; SCHRAGE, L. E. **A Guide to Simulation**. Springer- Verlag, , Second Edition. New York, 1987.

BROWN A. **Large-scale Component-Based Development**. Prentice Hall, 2000.

CAMERON, A. C.; TRIVEDI, P. K. **Regression Analysis of Count Data.** Cabridge University Press, New York, 1998.

CAMPOS, J. N. B.; NASCIMENTO, L. S. V.; STUDART, T. M. C. **Análise da Eficiência dos Métodos dos Momentos e da Máxima Verossimilhança na Estimativa de Parâmetros da Distribuição Gama II: Uma Abordagem Probabilística**. XV SIMPÓSIO BRASILEIRO DE RECURSOS HÍDRICOS, Curitiba, 2003.

CHAKRAVARTI, I. M.; LAHA, R. G.; ROY, J. **Handbook of Methods of Applied Statistics**. Volume I, John Wiley and Sons, 1967.

CHWIF, L. **Redução de modelos de simulação de eventos discretos na sua concepção: uma abordagem causal**. Tese de doutorado. Escola Politécnica da USP, Departamento de Engenharia Mecânica, 1999.

ÇINLAR, E. **Introduction to Stochastic Process**. Prentice Hall, EUA, 1975.

COOPER, R. B. **Introduction to Queuing Theory**. Oxford, New York, 1982.

CRNKOVIC, I. **Component-based Software Engineering – New Challenges in Software Developmen**t, 25th International Conference information Technology lnterfaces-IT/, June, Coratia, 2003.

DAI, J.; LIN, W.; MOORTHY, R.; TEO, C. **Berth Allocation Planning Optimization in Container Terminals.** Relatório Técnico, Georgia Institute of Technology and University of Singapore, 2003.

DEVROYE, L. **Non-uniform random variate generation**. Springer-Verlag, New York, 1986.

DUBOC, l.; ROSENBLUM, D. S.; WICKS, T. **A Framework for Modelling and Analysis of Software Systems Scalability**. Proceeding of the 28th International Conference on Software Engineering, ACM, Shangai, China, 2006.

ERLANG, A. K. **The Theory of Probabilities and Telephone Conversations.** Nyt ticlsskrift for Matematik, 1909.

FREITAS FILHO, P. J. **Introdução à Modelagem e Simulação de Sistemas com Aplicações Arena**. Visual Books, Florianópolis, 2001.

GAMBARDELLA, L. M.; RIZZOLI, A. E. **The role of Simulation and Optimisation in Intermodal Container Terminals**. European Simulation Symposium, Hamburg, Germany, 2000.

GENTLE, J. E. **Random Number Generation and Monte Carlo Methods**. Springer, Second Edition, New York, 2003.

GENTLE, J. E.; HÄRDLE, W.; MORI, Y. **Handbook of Computational Statistics – Concepts and Methods**. Springer, 2004

GERHARDT, I.; BARRY, L. N. **Transforming Renewal Process for Simulation of Nonstationary Arrival Process**. INFORMS Journal on Computing, April, 2009. Publicação online antes da versão impressa.

GIUNCHIGLIA, F.; VILLAFIORITA, A.; WALSH, T. **Theories of Abstraction**. Technical Report – MRG/DIST. Università di Genova. Nov, 1997. Disponível em: <https://eprints.kfupm.edu.sa/21662/1/21662.pdf>. Acesso em: 27 abril 2009.

GONZALES, J. A.; BOTTER, R. C. **Vantagens e Desvantagens da Aplicação de Técnicas de Simulação Versus a Teoria de Filas no Planejamento Portuário: Uma Discussão Conceitual e Uma Aplicação Prática**. Anais do 19o Congresso Nacional de Transportes Marítimos Construção Naval e Offshore, Rio de Janeiro, 2002.

HACKER, T.; SMITH, P. **Building a Network Simulation Model of the TeraGrid Network**. TeraGrid 2008 Conference, Las Vegas, NV, June, 2008.

HARROD, S.; KELTON, W. D. **Numerical Methods for Realizing Nonstationary Poisson Processes with Piecewise-Constant Instantaneous-Rate Functions**. Simultation: Transactions of The Society for Modeling and Simulation International, Vol 82, Nº 3, 2006, p.147–157.

HEISEY, S. A. **Determining Economic Efficiency in Harbors.** Depaepe-Willems Constest Submission, July, 2005.

HEYMAN, D. P.; SOBEL, M. J. **Stochastic Models in Operations Research: Stochastic Processes and Operating Characteristics**. Courier Dover Publications, 2004.

HOROWITZ, E.; SAHNI, S.; ANDERSON-FREED, S. **Fundamentals of Data Structures in C**, 2th ed., Silicon Press, 2007

ISO9126, **ISO/IEC 9126-1: Software Engineering -- Product Quality -- Part 1: Quality Model**. ISO, 2001.

KELTON, W. D.; SADOWSKI, R. P.; SADOWSKI, D. A. **Simulation with Arena**. Second Edition, McGraw-Hill, Boston, 2001.

KELTON, W. D. **Representing and generating uncertainty effectively**. Proceedings of the 2007 Winter Simulation Conference, USA, 2007.

KENDALL, D. G. **Stochastic Processes Occurring in the Theory of Queues and Their Analysis by the Method of Imbedded Markov Chains**. Annals Mathematical Statistics, Vol 24, September, 1953.

KENDALL, M.; STUART, A. **The Advanced Theory of Statistic, Vol 2: Inference and Relationship Theory.** Fourth Editon, Oxford University Press, New York, 1979.

KENDALL, M.; STUART, A.; ORD, J. K. **Kendall's Advanced Theory of Statistic, Volume 1: Distribution Theory**. Sixth Editon, Halsted Press, New York, 1994.

KNUTH, D. E. **The Art of Computer Programming, Volume 2: Seminumerical Algorithms**. Third Edition, Addison-Wesley, 1997.

KOLMOGOROV. N. **Grundgebriffe der Wahrscheinlichkeitsrechnung**. Berlin, Springer-Verlag, 1933.

KUHL, M. E.; STEIGNER, N. M.; LADA, E. K.; WAGNER, M. A.; WILSON J. R. **Introduction to Modeling and Generating Probabilistic Input Process for Simulation**. Proceedings of the 2008 Winter Simulation Conference, 2008, p. 48-61.

LAW, A. M.; KELTON, W. D. **Simulation Modeling Analysis**. Third Edition, McGraw Hill, 2000.

LEE, A. **Applied Queueing Theory**. MacMillan, New York, 1966.

L´ECUYER, P. **Good Parameters and Implementations for Combined Multiple Recursive Random Number Generators.** Operations Research, Vol. 47, Nº 1, Jan-Feb, 1999.

L´ECUYER, P.; SIMARD, R.; CHEN, E. J.; KELTON, W. D. **An Object-Oriented Random-Number Package With Many Long Streams And Substreams,** Operations Research, Vol. 50, Nº 6, December, 2002.

LEHMER, D. H. **Mathematical methods in large-scale computing units**. Annals of the Computation Laboratory of Harvard University, No. 26, Proceedings of a Second Symposium on Large-Scale Digital Calculating Machinery, 1951.

LEWIS, P.A.; SHEDLER, G. S. **Simulation of nonhomogeneous Poisson processes by thinning**. Naval Research Logistics Quarterly, Vol 26, New York, 1979, p. 403-413.

LUO, S.; MARIN, G. A. **Realistic Internet Traffic Simulation Through Mixture Modeling And A Case Study**. Proceedings of the 2005 Winter Simulation conference, 2005, p. 2408-2416.

MARSAGLIA, G.; ZAMAN, A.; MARSAGLIA, J. C. W. **Rapid evaluation of the inverse normal distribution function**. Statistic and Probability Letters, Vol. 19, 1994.

MASNIK FERREIRA, M. A. M.; ROSA, J. M. C; CARNIERI, C. **Improving Temporal Series Analysis Using an Iterative Algorithm for the Identification Stage**. WSEAS Transactions on Information Science & Applications, Issue 11, Volume 3, Greece, November 2006.

MASNIK FERREIRA, M. A.; MENDES JÚNIOR, R.; CARNIERI, C. **Análise de Desempenho de Sistemas Portuários Usando Simulação Matemática e Estatística**. Revista Produção On-line, Florianópolis, Vol. 7, N° 3, Novembro, 2007.

MATSUMOTO, M.; NISHIMURA, T. **Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator.** Vol. 8, Issue 1, ACM Transactions on Modeling and Computer Simulation, January, 1998.

McCALL, J. A.; RICHARDS, P. K.; WALTERS, G. F. **Factors in Software Quality**. Technical Report RADC-TR-77-369, US Department of Commerce, 1977.

MEYER, P. L. **Probabilidade: Aplicações à Estatística**. 2ª ed. Livros Técnicos e Científicos, Rio de Janeiro,1983.

MORETTIN, P. A.; TOLOI, C. M. C. **Análise de Séries Temporais**. Editora Edgard Blücher, São Paulo, 2004.

MUSSELMAN, K J. **"Guidelines for Success"**, In: **Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice**. J. Banks, John Wiley & Sons, New York, 1998, p. 721-743.

NANCE, R. E. **A History of Discrete Event Simulation Programming Languages.** In: *Proccedings of the Second ACM SIGPLAN History of Programming Languages Conferences*,1993, p. 149-175.

PAXSON, V.; FLOYD, S. **Wide Area Traffic: The Failure of Poisson Modeling**. IEEE/ACM Trans. on Networking, Vol. 3, N° 3, June, 1995, p. 226-244.

PRADO, D. **Teoria das Filas e da Simulação**. 2ª. Ed. INDG, Belo Horizonte, 2004.

PRESSMAN, R. S. **Software Engineering: a Practioner's Approach**. 6ª Ed, McGraw-Hill, 2004.

RICCI, V. **Fitting Distributions with R.** February, 2005. Disponível em: <http://cran.stat.sfu.ca/doc/contrib/Ricci-distributions-en.pdf>. Acesso em: 15 novembro 2007.

ROBINSON, S.; NANCE, R. E.; PAUL, R. J; PIDD, M.; TAYLOR, S. J. E. **Simulation Model Reuse: Definitions, Benefits and Obstacles**. Simulation Modelling Practice and Theory, Vol. 12, 2004, p. 479-494.

ROCKWELL SOFTWARE, **Arena Template Developer´s Guide**, Outubro, 2005.

SALIBY, E. **Repensando a Simulação**, São Paulo, Atlas, 1989.

SMIRNOV, N. V. **Sur la distribution de w^2**. C. R. Acad. Sci. Paris, 1936.

SNEDECOR, G. W.; COCHRAN, W. G. **Statistical Methods**. Eighth Edition, Iowa State, University Press, 1989.

TANENBAUM, A. S. **Computer Networks**. Fourth Edition, Prentice Hall, 2002.

TRIVEDI, K. S. **Probability and Statistics with Reliability Queuing and Computer Science Applications.** Second Edition. John Wiley & Sons, 2002.

VINCENT, S. **"Input Data Analysis"**, In: **Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice**. J. Banks, John Wiley & Sons, New York, 1998, p. 55-91.

VLIET, H. van. **Software Engineering: Principles and Practice.** John Wiley & Sons, 2007

VITHARANA, P. **Risks and Challenges of Component-Based Software Development**. Communications of the ACM, Vol. 46, Nº 8, August, 2003, p. 67-72.

YEGENOGLU, F.; FARIS, F.; QADAN, O. **A model for representing wide area internet packet behavior**, IEEE Proc. Int. Performance, Comp. Commun. Conf. (IPCCC), Phoenix, AZ, February, 2000, p. 167-173.